Contribution à l'étude et au développement de modèles connexionnistes séquentiels de l'apprentissage

Claude Touzet

Université de Montpellier II

THESE DE DOCTORAT

Contribution à l'étude et au développement de modèles connexionnistes séquentiels de l'apprentissage

Claude TOUZET

Soutenue le 14 décembre 1990, devant le Jury :

Prof. C. Durante, Université de Montpellier II

Prof. N. Giambiasi, Université d'Aix-Marseille III

Prof. J. Hérault, INP de Grenoble

Prof. C. Jutten, Université J. Fourrier; Grenoble

Prof. F. Prunet, Université de Montpellier II

O. Sébilleau, Ingénieur (DIGILOG, Aix-les-Milles)

SOMMAIRE

TABLE DES FIGURES

Pages :

INTRODUCTION GENERALE

Les réseaux de neurones artificiels, ou réseaux connexionnistes, sont dotés de propriétés, qui comme l'apprentissage à partir d'exemples, semblent prometteuses dans certains domaines d'applications. Il est d'usage de citer la reconnaissance de formes, le diagnostic, ... En fait, est considérée comme éligible tout problème qui se représente sous la forme d'une fonction de mise en correspondance entre un espace d'entrée et un espace de sortie, dès lors que l'on ne dispose que d'exemples de comportement de cette fonction.

Ainsi, dans l'application des techniques connexionnistes à un problème de diagnostic des douleurs abdominales en cas d'urgence [Le Cun 87a], l'espace d'entrée est constitué des symptômes, l'espace de sortie est composé des diagnostics possibles. A partir d'un ensemble de cas utilisés pour l'apprentissage, le réseau construit une fonction de mise en correspondance symptômes-diagnostic.

Jusqu'en 1985, les fonctions réalisables étaient limitées aux fonctions linéaires [Minsky 88]. Depuis, un grand progrés a été accompli avec l'algorithme d'apprentissage basé sur la rétro-propagation de gradient [Le Cun 87b], [Rumelhart 86] pour les réseaux multicouches. Celui-ci a permi d'aborder des problèmes non-linéaires dont l'exemple le plus représentatif est le ou-exclusif. E. Decamp et B. Amy recensent dans leur ouvrage [Decamp 88] des applications démonstratives de la puissance de cet algorithme.

Cependant, l'application immédiate des modèles connexionnistes, notamment ceux basés sur la rétropropagation de gradient, à des domaines tels que la reconnaissance de la parole [Watrous 89], nécessite la difficile prise en compte d'informations de nature séquentielle. Le problème réside dans l'utilisation d'un système combinatoire qui puisse traiter des informations temporelles.

Les approches proposées pour étendre le domaine d'application des réseaux neuronaux aux problèmes séquentiels se regroupent en quatre classes. Dans chaque cas, il y a ajout d'informations temporelles.

- La première utilise un recodage spatial des entrées temporelles. Les limitations découlent évidement du nombre restreint d'entrées pouvant être prise en compte. Le fonctionnement du réseau est combinatoire.
- La seconde approche fait appel à un codage de l'information temporelle localement sur le neurone. En fait, il y a ajout d'une boucle de rétroaction sur le neurone. Le comportement du réseau reste combinatoire, moyennant quelques aménagements des algorithmes d'apprentissage.
- La troisième classe regroupe les modèles dédiés à la résolution d'un problème particulier et par là même peu généraux.
- La dernière approche rassemble les réseaux bouclés. Ces modèles sont les plus efficaces. Nous distinguons au sein de ceux-ci les réseaux séquentiels, qui utilisent le concept d'état interne. Cette approche au vu des résultats déjà obtenus et des concepts manipulés, apparait la plus prometteuse pour résoudre des applications séquentielles. C'est dans ce cadre que nous avons développé le modèle de la machine séquentielle connexionniste.

___Ce modèle est une généralisation des modèles de réseaux séquentiels. A l'image de la machine séquentielle développée dans le cadre de la théorie des automates, il y découpage en deux blocs fonctionnels, la fonction de transition et la fonction de sortie. Chacune est réalisée par un réseau multicouches. L'apprentissage est basé sur l'algorithme de la rétropropagation de gradient modifié. Trois algorithmes d'apprentissage différents sont présentés. Les résultats obtenus sur de petits problèmes de reconnaissance de séquences mettent en avant certaines caractéristiques intéressantes par rapport aux machines séquentielles.

___Dans ce mémoire, nous distinguons trois chapitres. Le premier, aprés un bref rappel sur les réseaux de neurones artificiels, propose une classification des différentes approches connexionnistes du séquentiel. En particulier, nous décrivons les réseaux de neurones séquentiels. Puis, nous rappelons le modèle de la machine séquentielle et présentons succintement le modèle de la machine séquentielle connexionniste et les variantes de l'algorithme d'apprentissage. Nous concluons en regroupant les différents modèles de réseaux de neurones séquentiels présentés au sein du modèle plus général de la machine séquentielle connexionniste. Le second chapitre détaille la structure, le fonctionnement et les variantes d'apprentissage de ce modèle. Les propriétés émergentes sont illustrées sur de petits problèmes simples de détection de séquences. Le dernier chapitre décrit le logiciel SACREN, outil général de simulation de réseaux de neurones, développé et utilisé dans le cadre de ce travail.

CHAPITRE I

LES MODELES CONNEXIONNISTES ET LE SEQUENTIEL

1 Les réseaux de neurones artificiels

Dans cette première partie, nous rappellons rapidement quelques notions relatives aux réseaux de neurones artificiels. Ceci nous permet de préciser le vocabulaire utilisé tout au long du présent travail. Le lecteur désireux d'une introduction exhaustive aconsultera avec avantage [Karna 89].

Kohonen [Kohonen 88] propose la définition suivante : "Les réseaux de neurones artificiels sont des réseaux massivement connectés en parallèle d'éléments simples (habituellement adaptatifs) et leur organisation hiérarchique. Ils sont sensés interagir avec les objets du monde réel de la même manière que les systèmes nerveux biologiques."

Aujourd'hui, une manière simple de concevoir un réseau de neurones est de considérer qu'il s'agit d'un système de traitement de l'information composé d'un grand nombre de processeurs interconnectés (cellules). Chaque cellule calcule sa sortie sur la base des informations reçues des autres cellules qui lui sont connectées et des poids de ces connexions. Un réseau de neurones peut-être complètement décrit par la spécification des quatre éléments suivants :

- l'élément de traitement,
- la topologie du réseau (schéma d'interconnexions),
- le type d'apprentissage
- le schéma d'activation du système.

L'élément de traitement est décrit par une fonction de transfert, aussi appelée fonction d'activation, qui établie une correspondance entre les valeurs présentées en entrées de la cellule et la valeur de sortie. Dans la suite de notre propos, nous emploierons indifféremment les termes : élément de traitement, cellule, neurone, unité.

La topologie du réseau décrit le schéma des interconnexions entre les éléments de traitement. De nombreuses architectures de réseaux sont actuellement utilisées. L'architecture multicouches sans boucle est une des plus connues et des plus intéressantes parmi celles développées jusqu'ici.

L'algorithme d'apprentissage, habituellement dénommé "loi d'apprentissage" spécifie la manière dont les poids du réseau seront ajustés pour adapter le comportement du réseau à l'application.

Le schèma d'activation du système spécifie l'ordre de mise à jour des neurones du réseau. Il existe des modèles où les états des cellules sont calculés en parallèle, d'autres modèles recalculent l'état de chaque cellule en série d'après un tirage aléatoire, etc.

Les propriétés généralement mises en valeur des réseaux de neurones sont les suivantes :

. Apprentissage à partir d'exemples.
. Résistance aux pannes et au bruit.
. Calcul en parallèle, rapide.
. Adaptatif au changement de l'environnement.

En résumé, nous retiendrons que les réseaux de neurones sont constitués de simples éléments de calcul (cellules) communiquant entre eux par l'envoi de leur niveau d'activation au travers de connexions (synapses). La valeur de l'état des cellules est calculé par une fonction de transfert à partir des valeurs des entrées. Le comportement du réseau est déterminé par la topologie et les poids des connexions, les fonctions de transfert et le schéma d'activation. Les réseaux de neurones sont capables d'apprentissage à partir d'exemples.

En ne considérant que les modèles à apprentissage supervisé, un exemple est une forme d'entrée associée à une forme de sortie. La forme d'entrée est codée sur les cellules d'entrée sous la forme d'un vecteur d'activation. On associe souvent abusivement la forme d'entrée avec le vecteur d'activation des cellules de la couche d'entréee et réciproquement le vecteur d'activation des cellules de sortie avec la forme de sortie. Nous désignons par codage le passage de la forme au vecteur d'activation. L'opération inverse est le décodage.

1.1 Conventions et notations

Les lettres minuscucules en italiques représentent des quantités scalaires, x_i est la valeur de l'état du neurone i par exemple. Les lettres minuscules en gras représentent les vecteurs, $\mathbf{x} = (x_1, \ldots, x_n)$ est le vecteur d'état des neurones numéro 1 à n. Les lettres majuscules en gras représentent les matrices, **W** est la matrice des poids w_{ij}, où i indique la ligne et j la colonne.

Variables

x_i est l'état du neurone i.

$\mathbf{x} = (x_1, \ldots, x_n)$ est le vecteur d'état des neurones du réseau.

a_i est la somme pondérée des entrées du neurone i.

f est la fonction de transfert du neurone, $x_i = f(a_i)$.

w_{ij} est le poids de la connexion du neurone j vers le neurone i .

$\mathbf{i}_i = (e_1, \ldots e_k)$ est la forme d'entrée i.

$\mathbf{o}_i = (s_l, \ldots, s_n)$ est la forme de sortie i.

Graphiques

Neurone, cellule

Connexion orientée

Connexion de poids fixe (en gras)

2/ Quelques approches connexionnistes du séquentiel

Dans un circuit séquentiel, l'état des sorties à un instant donné dépend à la fois des entrées et de l'état des sorties à l'instant précédent. A n'importe quelle date, les sorties d'un circuit séquentiel sont fonction à la fois des entrées et de l'information stockée à cette date.

Soient $o(t)$ la sortie du réseau de neurones à la date t,

$i(t)$ l'entrée de ce réseau à la même date,

F la fonction réalisée par ce réseau,

alors, le comportement d'un réseau de neurones combinatoire est décrit par l'équation suivante :

$$o(t) = F(i(t)).$$

Le comportement d'un réseau de neurones séquentiel est décrit par :

$$o(t) = F(i(t), i(t-1), i(t-2), \ldots, i(1), i(0)).$$

Dans la littérature [Touzet 90b], nous recensons quatre groupes d'approches connexionistes utilisées pour traiter de problèmes séquentiels : les approches spatio-temporelles, les approches utilisant des boucles récurrentes sur le neurone lui-même, les approches dédiées, les approches utilisant des connexions récurrentes. Nous allons présenter et discuter rapidement des principes et des modèles de chacunes. La répartition des modèles de réseaux de neurones dans l'une ou l'autre de ces approches est réalisée sur des critères de nature structurelle. A partir de l'architecture prototypique, nous sommes en effet capable d'inférer un type de comportement face aux problèmes séquentiels et donc de connaitre les limitations intrinsèques. Dans le cas de l'approche spatio-temporelle, il n'y a pas de boucle dans le réseau. La seconde approche illustre l'existence de boucles de rétro-action sur le neurone lui-même. Sous le terme d'approche dédiée sont regroupées toutes les architectures particulièrement orientées vers la résolution d'un problème particulier. Enfin, les approches les plus générales regroupent les modèles utilisant des connexions récurrentes. Les plus intéressants de ces modèles, les réseaux de neurones séquentiels, sont présentés à part au paragraphe suivant (I.3).

2.1 Approche spatio-temporelle

Dans cette approche, une fenêtre regroupant plusieurs entrées précédentes permet de traiter les séquences, avec cependant plusieurs limitations. L'exemple le plus connu de ce type d'approche est le programme NETTalk [Sejnowski 86] représenté figure 1.1. Un réseau de neurones multi-couches doté de la rétro-propagation de gradient apprend à prononcer du texte

anglais. La prononciation de la lettre dépend des 3 lettres précédentes et des 3 lettres suivantes. La taille de la fenêtre est donc de 7 lettres. Une telle approche ne permet pas de prendre en compte plus de 3 lettres précédentes. Comme le note Sejnowski, atteindre un haut niveau de performance (contrôle de l'intonation, de la prosodie, ...) pour ce réseau nécessiterai la prise en compte d'une fenêtre plus large. Une entrée hors de la fenêtre ne peut pas intervenir sur la sortie du réseau, le passé lointain est inabordable par cette technique.

Dans ce type d'approche, le comportement est décrit par l'équation :

$o(t) = F(i(t+n), i(t+n-1), ..., i(t), i(t-1), ..., i(t-n))$, n fixé.

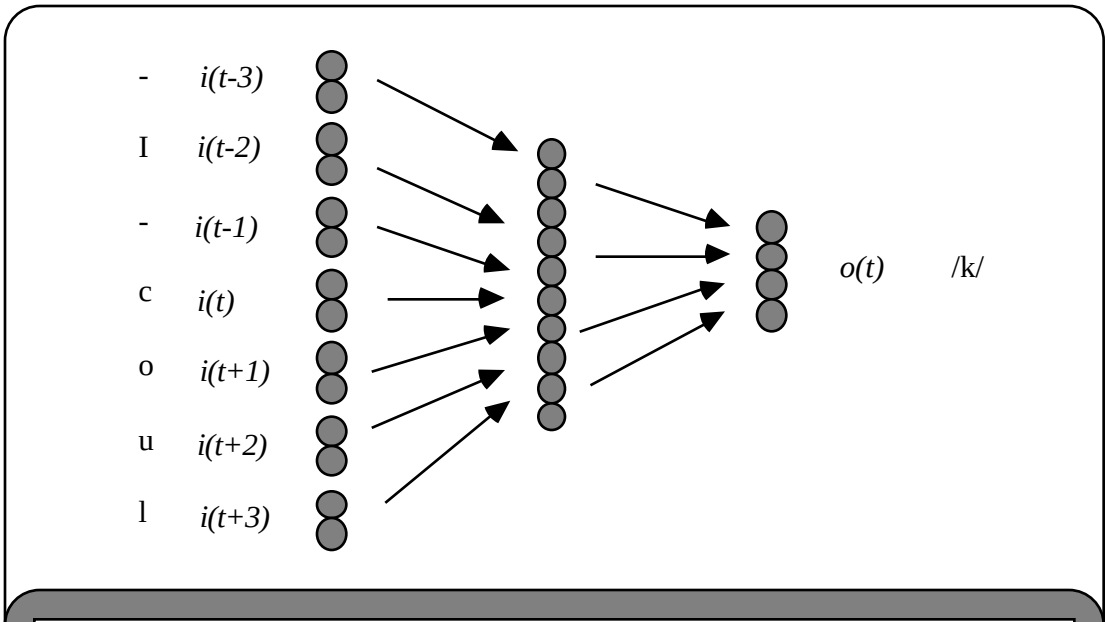

Figure 1.1 Schéma de l'architecture de NETtalk. Les entrées se répartissent en 7 groupes pour les 7 caractères traités à la fois. Un exemple de phrase est indiqué à gauche de la couche d'entrée. C'est le caractère du milieu qui est prononcé. La sortie correcte du réseau, dans ce cas, est indiquée à droite de la couche de sortie. La prononciation dépend des caractères précédants et futurs, limités ici à n = 7.

2.2 Approches utilisant des boucles récurrentes sur le neurone

Dans cette approche décrite figure 1.2, la prise en compte du passé est réalisée au niveau du neurone. La fonction d'activation est modifiée de façon à faire intervenir l'état passé pour le calcul de l'état présent. Le comportement est décrit par l'équation :

$o(t) = F(i(t), x(t-1))$, $x(t-1)$ est la valeur d'activation du neurone à la date précédente.

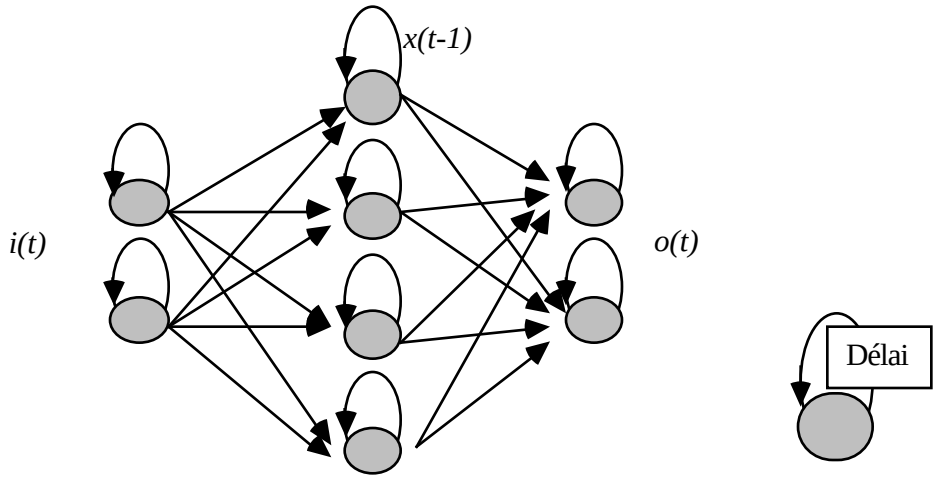

Figure 1.2. Approche utilisant des boucles récurrentes sur le neurone lui-même. La valeur d'activation du neurone dépend de sa valeur à la date précédente et de la somme pondérée de ses entrées.

Dans le modèle proposé par Schreter [Schreter 88], la valeur d'activation à la date t est utilisée pour le calcul de la valeur d'activation à t+1 :

$x(t+1) = x(t) + b.a(t).d(t).$

$a(t)$ et la somme pondérée des entrées, b est une constante.

$d(t) = 1-x(t)$ si $a(t) > 0$

$x(t)$ si $a(t) < 0$

0 si $a(t) = 0$

Le réseau est totalement connecté, les connexions sont inhibitrices, un neurone de contrôle est connecté à tous les autres neurones. L'apprentissage est de type hebbien [Hebb 49]. Dans ce modèle, deux types de mémorisation sont possible, la mémoire à long terme est représentée par des changements des poids des connexions et la mémoire à court terme représentée par les évolutions des valeurs d'activation des neurones.

Nous avons développé une idée semblable dans le cadre de réseaux multi-couches [Touzet 88]. Le neurone simule un phénomène de sommation spatio-temporelle : l'état de neurone à la date précédente est utilisé pour le calcul de son état présent. L'équation de la fonction d'activation est la suivante : $x(t+1) = f(a(t), x(t))$ où x est l'état du neurone, a est la somme pondérée des entrées, f une fonction de type sigmoïde.

$$x\ (t+1) = m\ .\ \frac{e^{(k\,a(t)+\partial.x(t))} - 1}{e^{(k.a\,(t)+\partial.x(t))} + 1}$$

m, k, ∂ sont des constantes.

Bien que ces modèles montrent des capacités certaines dans l'apprentissage et la reconnaissance de séquences, les performances notamment au niveau de la longueur des séquences sont faibles.

2.3 Approches dédiées

Sous le vocable d'approches dédiées se regroupe des réseaux dont l'architecture est particulièrement orientée vers la résolution d'un problème précis.

Ainsi dans le modèle biologiquement plausible proposé par Dehaene & al. [Dehaene 87], l'unité de traitement est composée de 3 neurones, les triades synaptiques construites pour appréhender les modifications à court terme d'efficacité synaptique grâce à une règle d'apprentissage locale de type Hebb. Celles-ci sont arrangées en réseau multicouches de grappes d'unités. Le réseau montre des capacités d'apprentissage et de reconnaissance de séquences.

Un second exemple est le modèle proposé par Kleinfeld [Kleinfeld 86] décrit figure 1.3. Deux ensembles de connexions concourent à doter le réseau de la capacité de générer des séquences. L'un stabilise le réseau dans l'état mémorisé actuel tandis que le second, dont l'action varie en fonction du temps, oblige le réseau à effectuer des transitions entre les états mémorisés.

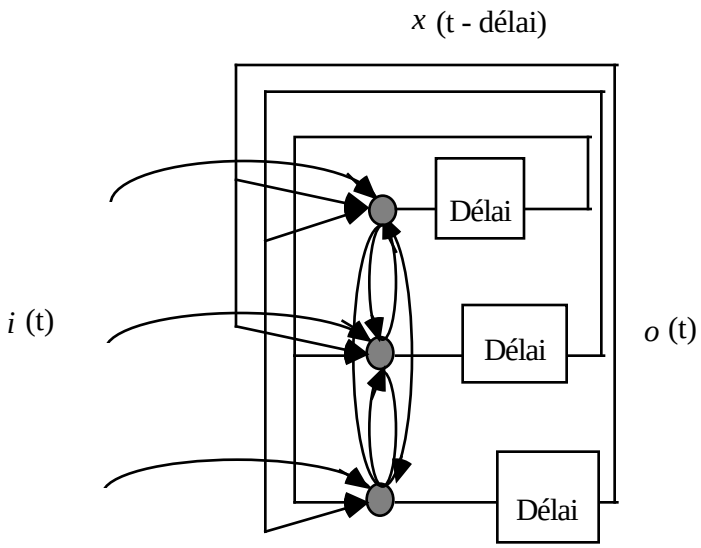

Figure 1.3. Architecture d'une mémoire associative génératrice de séquences.
Les synapses sans délai stabilisent le réseau dans les états stables.
Les synapses avec délai permettent la transition entre les états stables.
La reconnaissance des séquences est effectuée sur les entrées externes.

2.4 Approches utilisant des connexions récurrentes

Une première approche utilisée par Rumelhart & al. [Rumelhart 86] est décrite figure 1.4. Elle permet de construire pour chaque réseau à connexions récurrentes un réseau sans boucle avec le même comportement (sur une période de temps fini). L'algorithme d'apprentissage est la rétropropagation de gradient. Le principal problème rencontré est, à leur avis, la quantité de mémoire utilisée par l'algorithme d'apprentissage. En effet, la modification de poids d'une connexion doit être la somme des modifications pour chacunes des formes de la séquence. L'apprentissage est très lent requérant des milliers d'itérations, d'autre part la longueur des séquences est limitée, dans la pratique, à quelques unités.

Le comportement de ce réseau est décrit par l'équation :

$o(t) = F(i(t), i(t-1), ..., i(0))$

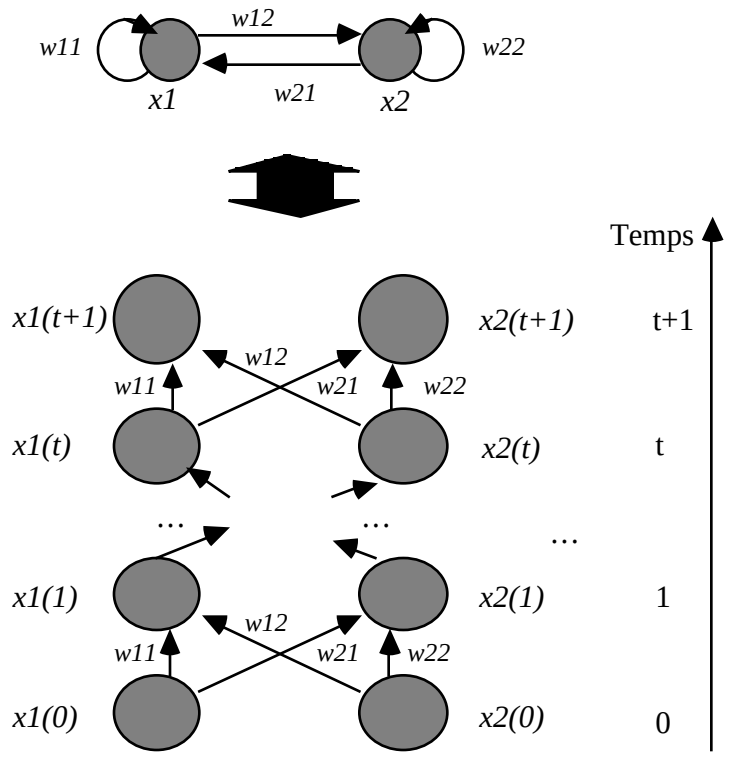

Figure 1.4. Un réseau récurrent et un réseau sans boucle ayant le même comportement. Les cellules sont dupliquées pour chaque unité de temps. Il est nécessaire que les poids connectant chaque couche dans le modèle sans boucle soient identiques pour toutes les couches. De plus, ces poids doivent être les mêmes que dans le réseau à connexions récurrentes.

C'est sur ce même principe que s'appuient les études visant à adapter la rétropropagation de gradient aux réseaux bouclés [Pineda 87], [Almeida 88], [Rohwer 88], [Williams 88], [Pineda 89], [Gori 89].

Kohonen s'est intéressé à la production de séquences par une mémoire auto-associative [Kohonen 87a]. Il utilise pour ce faire un modèle ayant l'apparence de la machine d'état fini, décrit figure 1.5. Il y a 3 groupes d'entrées sur la mémoire, deux pour les entrées primaires (forme d'entrée et contexte) et une pour le rebouclage de la sortie. Les possibilités de ce modèle sont, comme celles des mémoires associatives, limitées en ce qui concerne les problèmes non linéaires.

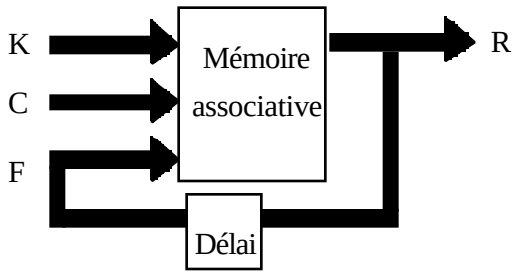

Figure 1.5. La mémoire associative de Kohonen pour la reconnaisance de séquences.
Elle a l'apparence de la machine d'état fini.
L'entrée K est l'entrée primaire de la forme.
C est le contexte relié à une séquence particulière.
F est le rebouclage des sorties.

3 Les réseaux de neurones séquentiels

Dans le cadre des approches utilisant des connexions récurrentes, nous avons choisi de présenter à part les réseaux de neurones séquentiels. La principale différence est qu'ils font appel à une représentation interne du contexte historique. Ces modèles montrent de bonnes performances en apprentissage de séquences et sont suffisamment généraux pour traiter une grande classe de problèmes séquentiels.

Nous allons présenter deux modèles parmis les plus renommés de ces réseaux : le modèle de Jordan et le modèle d'Elman. Il existe d'autres modèles du même type [Allen 88], [Jordan 88]. Leurs architectures multicouches permettent de réaliser des associations complexes. Ils sont adaptés à la résolution de problèmes séquentiels du fait de connexions récurrentes. Ils sont faciles d'emploi et couvrent un vaste champ d'application gràce à un apprentissage basé sur l'algorithme de la rétropropagation de gradient. Depuis, ces modèles ont été repris par plusieurs auteurs [Pearlmutter 88], [Smith 89], [Tsung 89], [Liu 90]. D'autres auteurs seront cités avec leurs expérimentations dans la suite de notre propos.

3.1 Le modèle de Jordan

Ce modèle, aussi appelé "state network" [Jordan 86], a été le premier modèle de réseau récurrent à faire apparaitre l'idée d'une fonction de transition et d'une fonction de sortie. Ce réseau apprend à générer des séquences gràce à un algorithme dérivé de la rétropropagation de gradient. Ce modèle est testé sur le problème de la coarticulation en prononciation de la parole. Les performances restent modestes du fait des limitations sur la fonction de transition.

L'architecture du réseau de Jordan est présentée figure 1.6. Les cellules d'entrée se répartissent en deux groupes : les cellules de plan et les cellules d'état. Les connexions entre cellules d'état et cellules cachées sont totales, de même pour la connectivité des cellules de plan avec les cellules cachées et des cellules cachées vers les cellules de sortie.
Les cellules de sortie sont rebouclées sur les cellules d'état par des connexions de poids fixes, de même pour les cellules d'état qui rebouclent sur elle-même par des connexions de poids fixes.

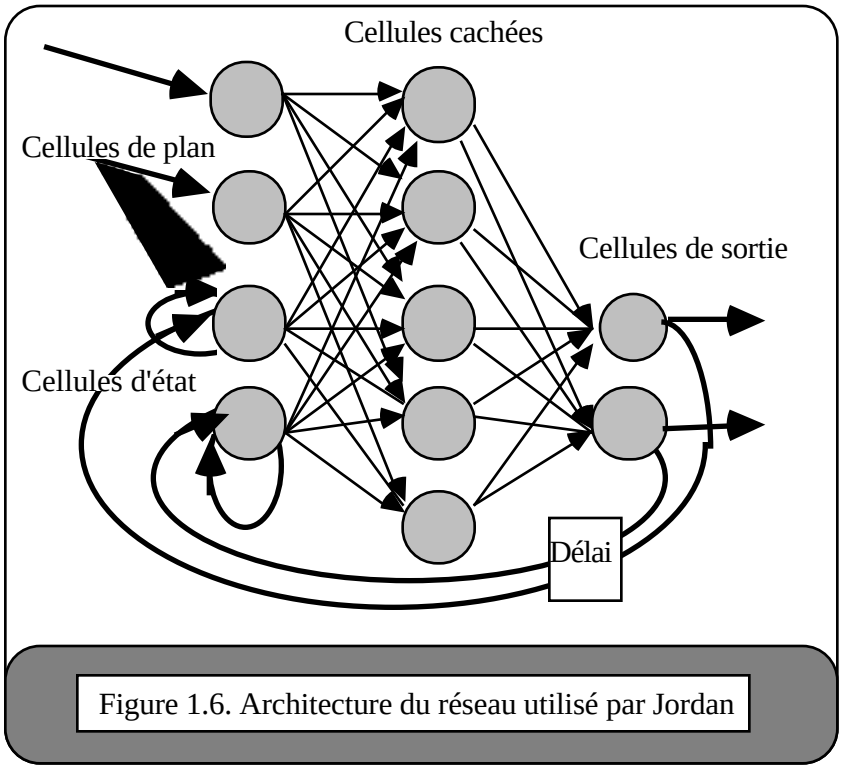

Figure 1.6. Architecture du réseau utilisé par Jordan

Le comportement de ce modèle est donné par l'équation : $o(t) = F(i, o(t-1))$

Ce réseau est capable de générer des séquences. Un vecteur d'activation *i* est appliqué sur les cellules de plan. Ce vecteur ne sera plus modifié pour la suite de la production de la séquence. Les vecteurs d'activation des cellules de sortie varient dans le temps du fait des connexions récurrentes sur les cellules d'états. Ces connexions modifient les valeurs d'activation des cellules d'état et imposent une entrée variable dans le temps au réseau multicouches. Le vecteur d'activation des cellules de plan restent constant pour une même séquence. En utilisant des vecteurs d'activation différents pour les cellules de plan, le même réseau apprend plusieurs séquences différentes.

Les connexions récurrentes sont de poids fixes, égaux à 1 pour les connexions depuis la couche de sortie et égaux à 0.5 pour les connexions récurrentes des cellules d'état.

Ainsi que l'expose l'auteur, ce système représente explicitement le contexte temporel sous la forme d'un vecteur d'état (aussi appellé état interne). La sortie est déterminée grâce à une fonction de sortie. A chaque instant, la sortie est calculée sur la base de l'état interne. L'état interne est remis à jour pour la prochaine sortie. Deux fonctions sont donc calculées à chaque instant, la fonction de sortie et la fonction de transition.

La fonction de sortie émerge de l'apprentissage des associations vecteur de plan, vecteur d'état avec le vecteur de sortie. L'algorithme employé est une généralisation de la

rétropropagation de gradient : plutôt que d'utiliser une valeur fixe pour les sorties désirées, des contraintes sont imposées sur ces valeurs. Une contrainte spécifie, par exemple, un intervalle pour la valeur de la sortie. Il y a apprentissage après chaque forme produite.

La fonction de transition est figée : les poids des connexions récurrentes sont fixés.

Quelques simulations démonstratives ont été réalisées dans le domaine de la production de la parole. Par exemple, un réseau composé de 8 cellules d'état, 6 cellules de plan, 10 cellules cachées, 8 cellules de sortie, est capable d'apprendre à générer une séquence de 8 éléments, chaque élément est constitué de 8 caractéristiques. On note que le réseau est capable d'anticiper sur les formes à venir.

De l'avis de son auteur, il n'est pas indispensable que la fonction de transition émerge de l'apprentissage, comme le fait la fonction de sortie, pour que le système soit capable de produire des séquences. Il ajoute que peu de puissance de calcul est perdue par une fonction de transition fixée. Comme le remarquent D. Servan-Schreiber et al. [Servan-Schreiber 88], ce réseau sait utiliser l'état interne, mais il ne sait pas le représenter.

3.2 Le modèle d'Elman

Ce modèle, aussi appelé "context network" [Elman 88], propose une architecture légèrement différente de celle de Jordan. Ainsi, la couche cachée a la tâche d'établir à la fois les associations de la fonction de sortie et de la fonction de transition. L'algorithme d'apprentissage reste la rétropropagation de gradient. Le principal résultat est que le réseau développe des représentations internes significatives du problème séquentiel soumis. D. Servan Schreiber et al. [Servan-Schreiber 88] ont tenté de décrire et de comprendre quel type de représentation se développe dans ce modèle, dans le cas particulier de l'apprentissage d'une grammaire d'état fini.

Comme l'illustre la figure 1.7, le réseau est à 3 couches.

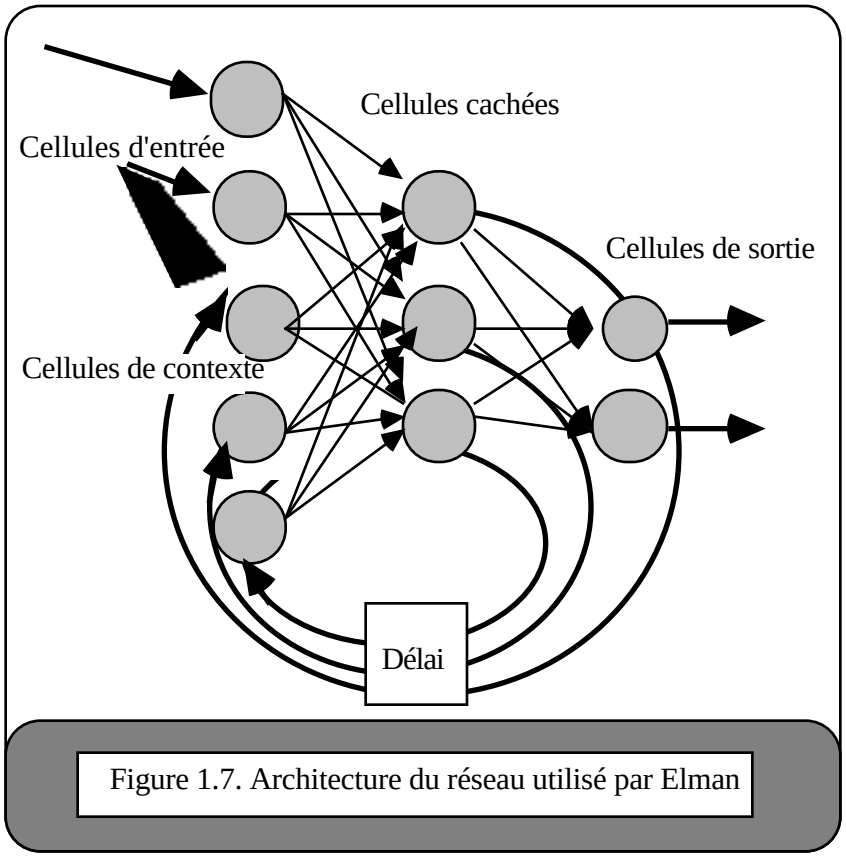

Figure 1.7. Architecture du réseau utilisé par Elman

La couche d'entrée consiste en deux ensembles de cellules, les cellules de contexte et les cellules d'entrée. Il y a autant de cellules cachées que de cellules de contexte. Chaque cellule cachée est reliée à une seule cellule de contexte par une connexion fixe, de poids égal à 1.

La tâche assignée au système est la prédiction de l'élément suivant de la séquence. A chaque instant, une sortie est calculée sur la base de l'état interne (contexte) et de l'entrée présente. Il y a copie du vecteur d'activation de la couche cachée sur la couche de contexte. De cette façon, le résultat intermédiaire du traitement à la date précédente (t-1) peut infuencer le traitement à la date actuelle (t).

L'algorithme d'apprentissage est la rétropropagation de gradient. Il faut remarquer le grand nombre d'itérations d'apprentissage nécessaires. Par exemple, dans le cas étudié par D. Servan-Schreiber et al., la base d'exemples est composée de 200000 séquences, générées en accord avec la grammaire d'état fini de Reber. Le graphe d'état de cette grammaire est composé de 5 états, 10 transitions, 7 valeurs d'entrées. Le réseau de neurones employé se compose d'une couche de 7 cellules d'entrée, 3 cellules de contexte, 3 cellules cachées et 7 cellules de sortie.

Dans leurs conclusions, ces auteurs évoquent les différentes étapes suivantes lors de l'apprentissage :

- Initialement, le réseau apprend à distinguer entre les entrées indépendamment du contexte temporel. L'information contenue dans la couche contexte est ignorée à ce moment.
- A la fin de cette étape, chaque entrée est associée avec un vecteur d'activation de la couche cachée spécifique.
- Dans la phase suivante, les différentes occurences de la même entrée sont distinguées sur la base des entrées immédiatement précédentes.
- Au fur et à mesure, la distinction implique des entrées précédentes de plus en plus nombreuses. Il est possible d'obtenir de cette manière et sous les conditions appropriées des représentations internes codant la séquence entière des éléments présentés.

Lorsque les exemples d'apprentissage sont issus d'une grammaire à états finis, ce modèle peut être utilisé comme l'automate à états finis correspondant. La représentation interne converge vers les noeuds de la grammaire. Les performances sont fortement liées à la taille de la grammaire [Smith 89].

3.3 Conclusion

Le modèle de Jordan fait apparaitre les notions nouvelles dans le cadre des études neuromimétiques de fonction de transition et fonction de sortie. Ce modèle utilise le concept d'état interne, bien qu'il ne sache pas le représenter.

Le modèle d'Elman, plus récent, utilise ces deux notions. Il développe, de plus, une représentation des états internes cohérente du problème soumis.

___Il est intéressant de remarquer que ces deux approches introduisent des notions et des concepts depuis longtemps manipulés par les automaticiens pour la description des circuits séquentiels. Ce sera l'objet du prochain chapitre.

4/ Rappel : le modèle de la machine séquentielle

Dans cette partie, nous présentons le modèle de la machine séquentielle développé par la théorie des automates pour modéliser les circuits séquentiels [Kohavi 78], [Hartmanis 66].

4.1 Différences entre circuit séquentiel et circuit combinatoire

Les principales différences entre circuit combinatoire et circuit séquentiel sont, pour les circuits séquentiels :

- la présence d'éléments mémoires tels que les bascules,
- la sortie est dépendante des entrées présentes et des entrées précédentes,
- le comportement est décrit par un ensemble de fonctions de sortie et de fonctions d'état suivant.

4.2 Définition et description

Un modèle de représentation des circuits séquentiels réels a été développé par la théorie des automates : la machine séquentielle représentée figure 1.8.

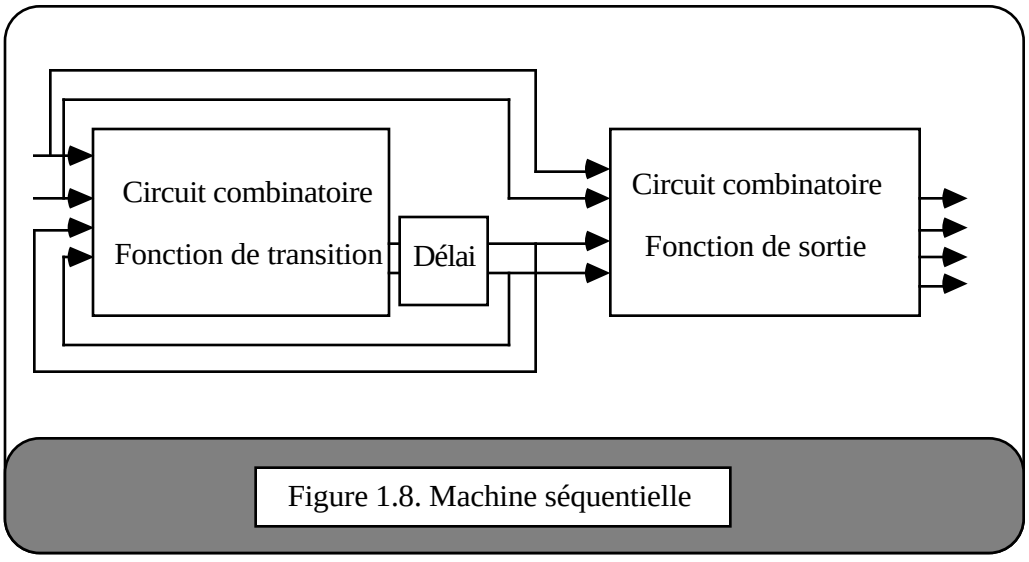

Figure 1.8. Machine séquentielle

Définition : une machine séquentielle est décrite par le quintuplet

$$M = (I, O, S, d, l)$$

où I, O, S sont respectivement les ensembles non vides des entrées, des sorties et des états.

d : I X S -> S est la fonction de transition.

l : I X S -> O est la fonction de sortie.

La fonction de transition δ fait évoluer l'état interne de la machine en fonction des entrées et du précédent état interne. La fonction de sortie λ à partir de l'état interne et de l'entrée calcule la sortie correspondante. Ainsi pour un même état interne, il peut y avoir des sorties différentes en fonction de l'entrée et réciproquement, pour des états internes différents la même sortie. La structure de la machine séquentielle explicite le rôle de la fonction de transition et de la fonction de sortie.

Le modèle de la machine séquentielle montre un comportement synchrone ou asynchrone. On parle de comportement synchrone lorsque la prise en compte des entrées est synchronisée avec un signal d'horloge externe. Un comportement est dit asynchrone lorsque les variations des entrées sont prises en compte dès leur arrivée.

4.3 Choix pour notre étude

Dans notre étude, nous choisissons un comportement synchrone pour le modèle. Nous choisissons aussi que l'horloge du système soit synchronisée sur la présentation des entrées. Nous choisissons encore que la présentation de l'entrée $i(t)$ au modèle soit synchronisée avec celle de $s(t-1)$.

Ceci s'exprime aussi de la façon suivante :

Soient les formes $i(t)$, $o(t)$, $s(t-1)$, $s(t)$ appartenant respectivement aux ensembles I, O, S, S, alors :

$$d\ (i(t),\ s(t-1)) = s(t).$$

$$l\ (i(t),\ s(t-1)) = o(t).$$

où t est un indice qui note la position dans la séquence.

5/ La machine séquentielle connexionniste

Nous utilisons l'acquis du domaine des circuits séquentiels pour proposer un modèle de réseaux de neurones traitant de problèmes séquentiels. La machine séquentielle connexionniste reprend le modèle de la machine séquentielle, en remplaçant les circuits combinatoires par des réseaux de neurones multicouches. Ces réseaux sont ceux qui, à l'heure actuelle, présentent les meilleures potentialités en ce qui concerne le traitement des problèmes non-linéaires.

Nous allons exposer rapidement la structure et le fonctionnement de la machine séquentielle connexionniste. Nous y reviendrons plus en détails au chapitre II.

5.1 Structure

Conformément au modèle classique de la machine séquentielle, une machine séquentielle connexionniste (figure 1.9) comporte deux sous-parties :

- l'une réalise la fonction de transition,
- l'autre réalise la fonction de sortie.

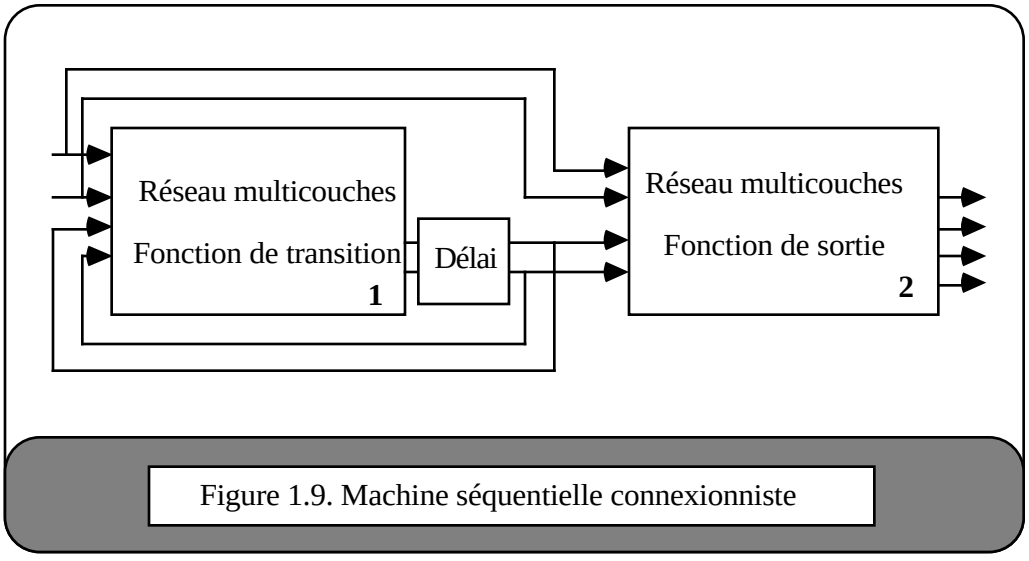

Figure 1.9. Machine séquentielle connexionniste

Chacune de ces deux fonctions est représentée par un bloc sur la figure 1. Le bloc 1 réalise la fonction de transition et le bloc 2 la fonction de sortie. Chaque bloc est un réseau de neurones multi-couches. Le bloc 1 reçoit les entrées primaires, ainsi que des boucles de rétro-actions depuis sa sortie. Le bloc 2 reçoit lui aussi les entrées primaires auquelles s'ajoute la sortie du bloc 1.

La figure 1.10 détaille l'architecture neuronale de la machine séquentielle connexionniste. Les neurones qui composent cette structure sont tous identiques. Par contre, les connexions se répartissent en deux classes.

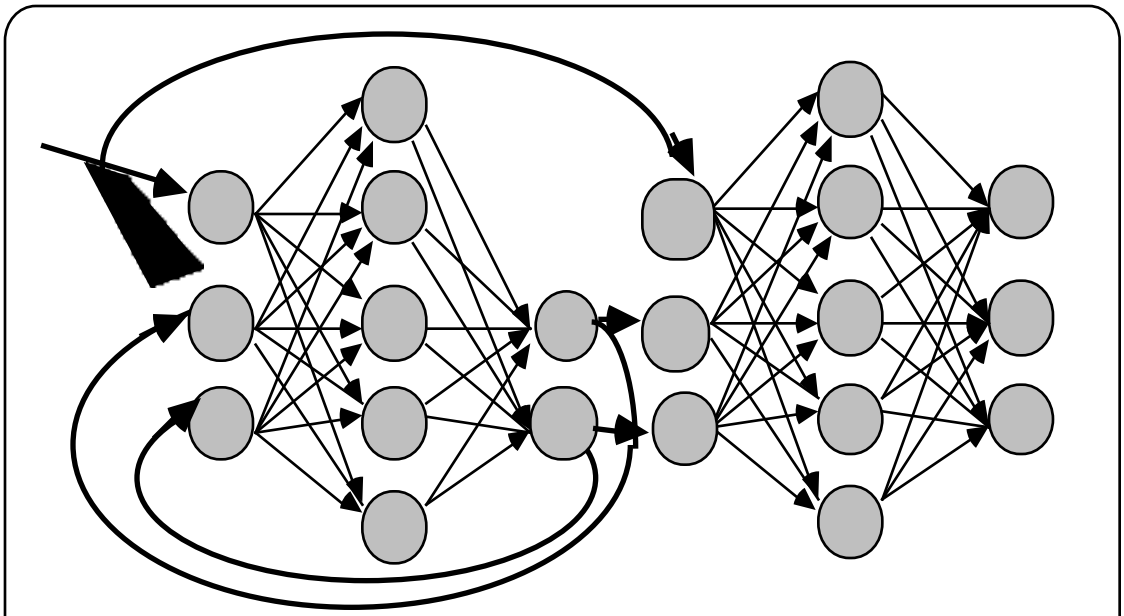

Figure 1.10. Architecture neuronale de la machine séquentielle connexionniste

Les connexions en provenance des entrées extérieures et celles en provenance de la sortie du réseau de transition sont de poids fixes. Elles sont représentées en traits gras. Leurs poids ne sont pas modifiés lors de l'apprentissage. Ces connexions sont univoques : chaque neurone de sortie est connecté à un seul neurone d'entrée. Ce neurone est appelé neurone d'entrée secondaire. Les connexions en provenance des entrées extérieures se font sur les neurones d'entrées primaires.

Les connexions entre les différentes couches dans chaque réseau sont plastiques, leur poids est modifié durant l'apprentissage. Tous les neurones d'une couche sont connectés à tous les neurones de la couche suivante. Il n'y a pas de connexions entre neurones d'une même couche, ainsi que de connexion récurrente sur le neurone lui-même.

5.2 Fonctionnement

Le réseau de transition calcule à partir de l'état présent et de l'entrée extérieure l'état interne. Cet état interne est le prochain état présent. Le réseau de sortie calcule à partir de l'état présent

et de la même entrée extérieure la sortie du modèle. Le fonctionnement est rappelé sur la figure 1.11.

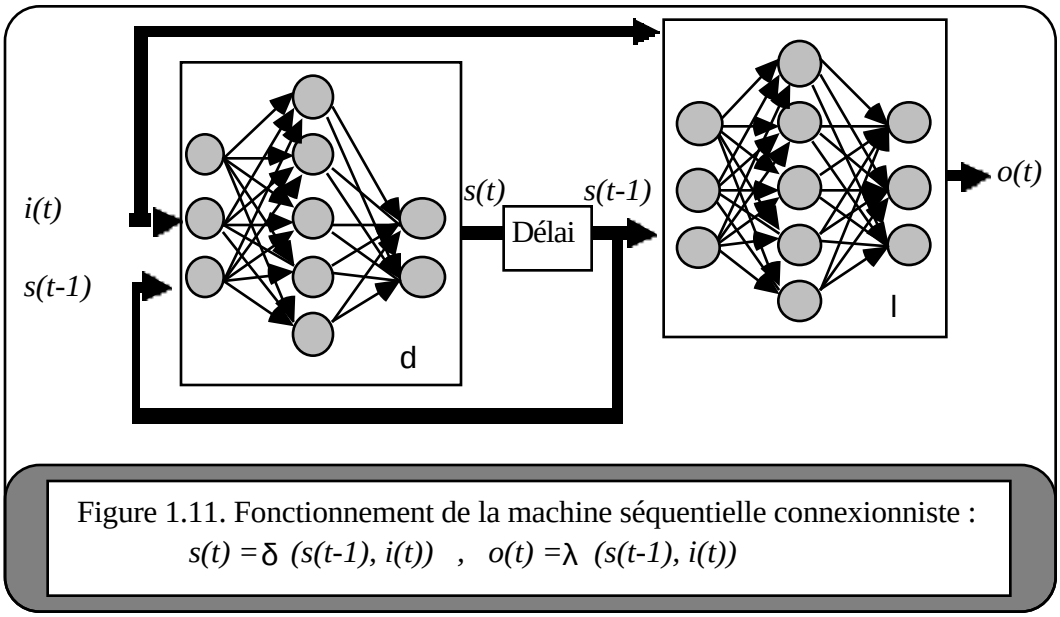

Figure 1.11. Fonctionnement de la machine séquentielle connexionniste :
$s(t) = \delta\ (s(t-1), i(t))$, $o(t) = \lambda\ (s(t-1), i(t))$

Le comportement de ce modèle est celui d'une machine séquentielle déterministe synchrone. Toutes les modifications d'état des entrées primaires et secondaires ne sont prises en compte qu'à certaines dates précises, fixées par une horloge. En dehors de ces instants particuliers, le modèle est bloqué car la boucle de rétro-action est ouverte. Celle-ci ne se ferme qu'à chaque instant d'horloge et permet alors la remise à jour de l'état présent.

5.3 Conclusion

A la différence des réseaux séquentiels décrits au paragraphe 3, ici la séparation des fonctions entre fonction de transition et fonction de sortie est clairement réalisée. En avant-propos du paragraphe 7, nous pouvons déjà annoncer la possibilité de réaliser l'apprentissage sur une seule fonction au choix.

6/ Synthèse des systèmes séquentiels connexionnistes

A partir de la procédure générale de synthèse des systèmes séquentiels synchrones, nous explicitons les modifications apportées par l'utilisation de la machine séquentielle connexionniste. Trois variantes de l'apprentissage sont présentées. Il s'agit de l'apprentissage à états supervisé, de l'apprentissage à états contraints et de de l'apprentissage à états non supervisés.

6.1 Utilisation de la machine séquentielle

La procédure générale de synthèse est présentée figure 1.12. Il faut remarquer qu'une importante limitation de ce modèle est relative au nombre d'états internes qu'il est possible de coder. Ce nombre est directement lié à la capacité de stockage de la machine et influence la longueur des séquences qu'il est possible de traiter.

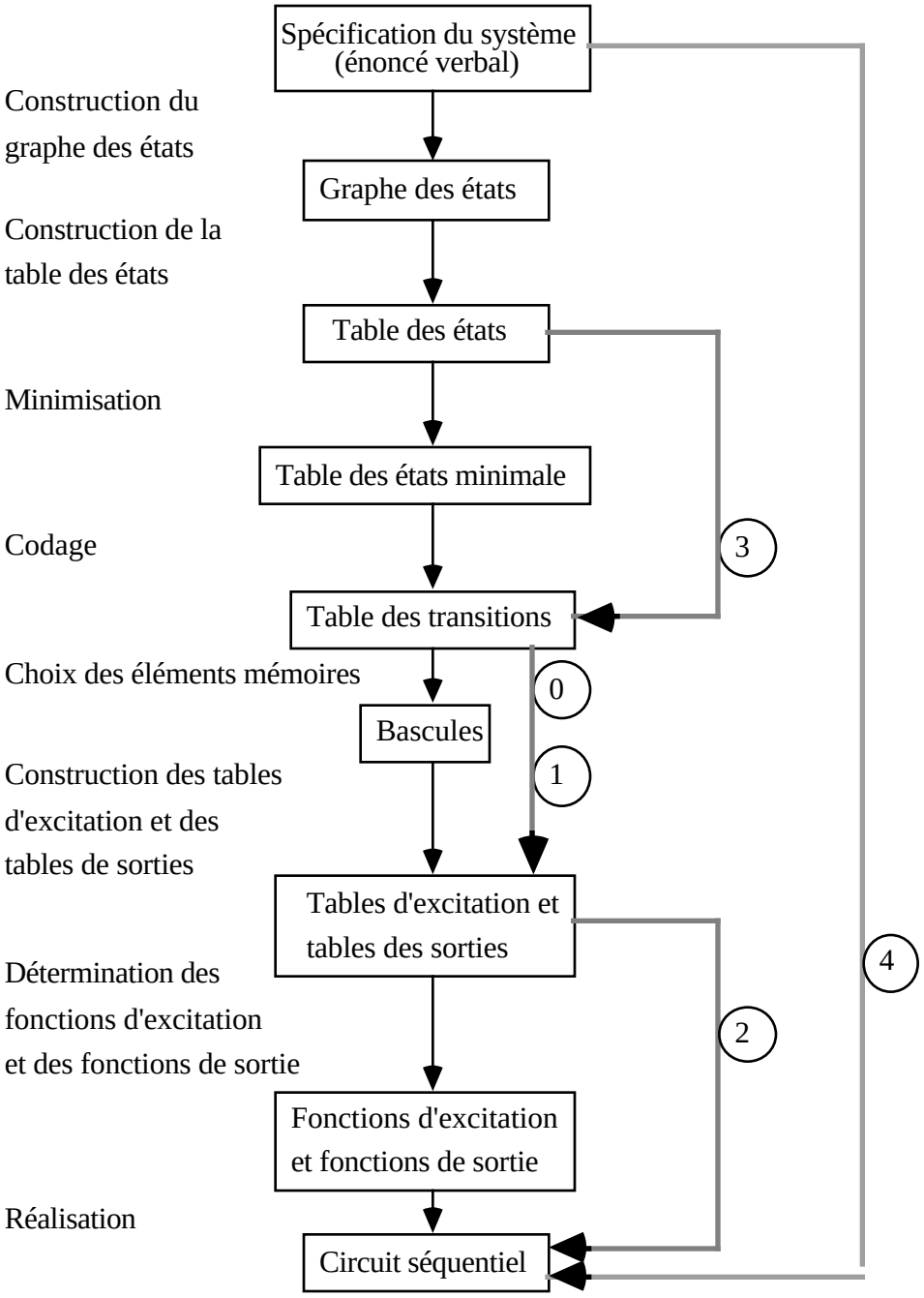

Figure 1.12. Procédures générales de synthèse des systèmes séquentiels synchrones avec les modèles de la machine séquentielle (⟶), de la machine séquentielle connexionniste à apprentissage à états supervisés (0,1,2), à apprentissage à états contraints (0,1,3), à apprentissage à états non supervisés (0,1,4)

6.2 Utilisation de la machine séquentielle connexionniste

6.2.1 A apprentissage à états supervisés

Elle introduit les modifications numérotées 0, 1 et 2 par rapport à la procédure originale de synthèse :

0 : Les éléments de délai (bascules ou retards) sont placés sur les connexions récurrentes de poids égal à 1, dont la fermeture est synchronisée sur le signal d'horloge. Ces connexions réalisent une recopie de l'état des neurones de sortie sur les neurones d'entrées secondaires. De ce fait, il y a identité entre la table des transitions et les tables d'excitation et de sortie.

1 : L'emploi de réseaux de neurones artificiels impose de définir à ce niveau un ensemble de séquences d'apprentissage.

2 : La détermination des fonctions d'excitation et de sortie est réalisée pendant la phase d'apprentissage. Le réseau est vu ici comme un circuit uniquement combinatoire. Nous choisissons comme algorithme d'apprentissage la rétropropagation de gradient. Les séquences d'apprentissage sont celles définies en 1.

Par rapport à la machine séquentielle, ce modèle propose la détermination automatique des fonctions d'excitation et de sortie. D'autre part, il est possible de généraliser le traitement à des séquences d'entrées à valeurs discrètes, même si l'apprentissage n'a été réalisé que sur des entrées à valeurs binaires.

6.2.2 A apprentissage à états contraints

Cet algorithme d'apprentissage introduit les modifications 0, 1, 2 et 3.

0, 1, 2 : Voir ci-dessus.

3 : La minimisation de la table des états, ainsi que le codage des états sont assurés en partie par l'utilisation d'intervalles de valeurs sur les cellules de sortie des réseaux réalisant la fonction de transition et la fonction de sortie.

Les résultats montrent la possibilité de minimisation automatique de la table des états. La définition du codage est relativement moins contraignante pour le réseau que dans le cas précédent. A cela, il faut ajouter la multiplication des états par la machine qui permettent de retracer la succession des entrées conduisant à l'état présent.

6.2.3 A apprentissage à états non supervisés

Cet algorithme introduit les modifications 0, 1 et 4.

0, 1 : Voir ci-dessus.

4 : Cet algorithme, à partir des séquences d'apprentissage définies en 1, essaye de construire la fonction de sortie correspondante. Il fait appel à la rétropropagation de gradient. Si le problème est réellement de nature séquentielle, alors il faut modifier la fonction de transition. Précédement à la modification de la fonction de transition par la rétropropagation de gradient, un algorithme de rétropropagation de la valeur désirée sur la fonction de sortie est employé.

Ce modèle effectue, à partir d'un ensemble de séquences, une synthèse automatique de l'automate correspondant. Il détermine la nature séquentielle ou combinatoire d'un problème. Il utilise les propriétés de généralisation des réseaux neuronaux pour traiter des séquences non apprises.

6.2.4 Conclusion

Les trois modèles présentés ont en commun la réalisation automatique des fonctions d'excitation et de sortie. L'ordre des présentations adopté montre une réduction des connaissances nécessaires (graphe des états, minimisation, ...) pour la résolution du problème en contre-partie d'une complexification de l'algorithme d'apprentissage. Chaque modèle est repris au chapitre suivant et dispose de son propre domaine d'application.

7 Algorithmes d'apprentissage pour la machine séquentielle connexionniste

Nous avons étudié différents types d'apprentissage sur le modèle de la machine séquentielle connexionniste : l'apprentissage à états supervisés, l'apprentissage à états contraints et l'apprentissage à états non supervisés.

7.1 Apprentissage à états supervisés (figure 1.13)

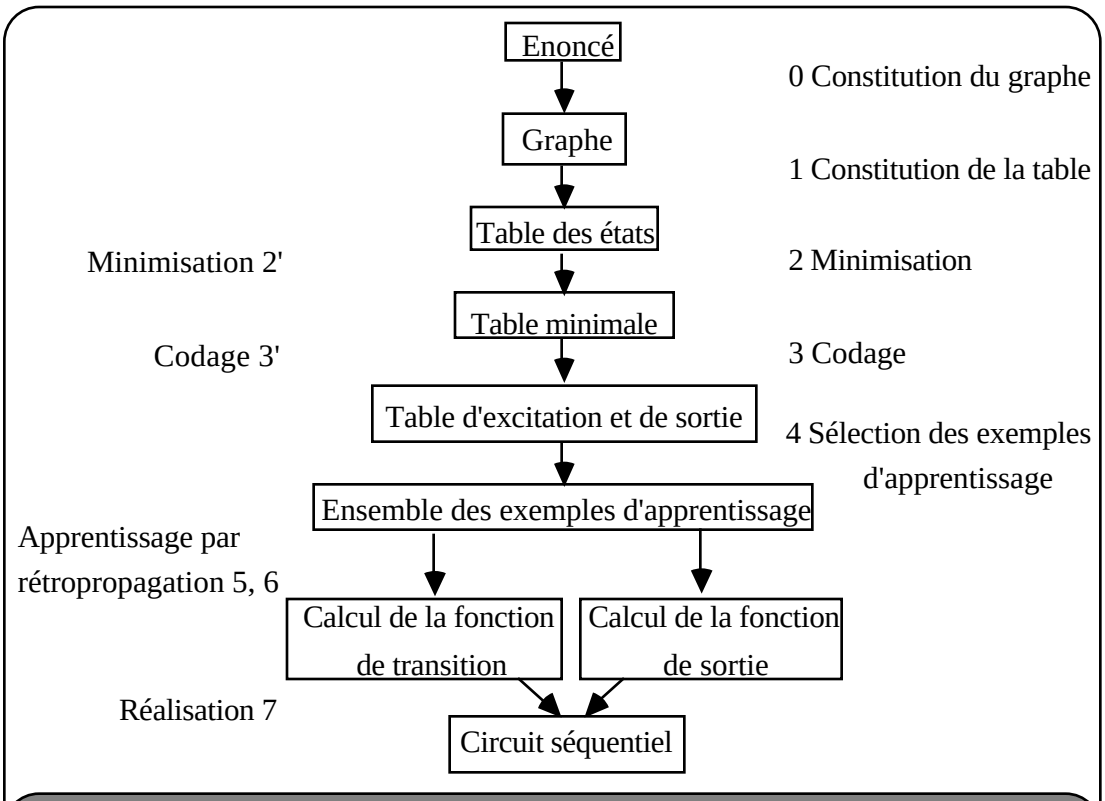

Figure 1.13. Procédure d'utilisation de la machine séquentielle connexionniste à apprentissage à états supervisés (0, 1, 2, 3, 4, 5, 6, 7) et à apprentissage à états contraints (0, 1, 2', 3', 4, 5, 6, 7). A gauche, les actions réalisées par le réseau. A droite, celles réalisées par l'utilisateur.

0, 1, 2 : Ce sont les mêmes opérations à mettre en oeuvre que dans le cas de la machine séquentielle. Le superviseur doit connaître le graphe d'états de l'automate à synthétiser.

3 : Il s'agit de définir un codage des formes d'entrée, des états internes et des formes de sortie sur le réseau.

4 : Il s'agit de sélectionner par rapport au graphe des états un ensemble de séquences d'apprentissage représentatif, qui permette au réseau de réaliser des associations correctes au niveau des deux procédures suivantes (5 et 6). Les exemples utilisés pour l'apprentissage à états supervisés sont de la forme suivante :

$\{[$(Entrée, Etat présent) -> (Etat interne désiré, Sortie désirée)$]_j\}_i$

pour la forme j de la séquence i.

5, 6 : La machine est composée de deux blocs fonctionnels. Il y a une phase d'apprentissage pour la fonction de transition d'état et une phase d'apprentissage pour la fonction de sortie. Il y a une itération d'apprentissage après chaque présentation d'une forme de la séquence et non pas seulement à la fin de la séquence. L'apprentissage au sein de chacun des blocs fonctionnels est réalisé à l'aide de l'algorithme de la rétropropagation de gradient. Il peut être réalisé en parallèle sur chacun des blocs. Afin de pouvoir appliquer l'algorithme, il est nécessaire de pouvoir définir un signal d'erreur pour chaque bloc.

5 : Apprentissage de la fonction de transition d'état

Le codage des états internes fourni les données nécessaires à l'établissement d'un signal d'erreur sur la couche de sortie du bloc transition d'état. Ce codage est choisi par le superviseur. Il justifie le terme d'apprentissage à états supervisés.

6 : Apprentissage de la fonction de sortie

Le signal d'erreur est la différence entre la valeur désirée et la valeur calculée par propagation des neurones de sortie.

Conclusion

Nous montrons que la généralisation à des séquences de valeurs discrètes, autres que binaires, est réalisable. Par exemple, ayant appris à se comporter comme un automate détecteur de la séquence binaire 010, la machine séquentielle connexionniste est capable de reconnaitre une séquence telle que : 0.8 0.1 0.9 comme bonne. Le système n'est plus à la recherche de la séquence binaire 010, mais d'une variation des entrées du type : valeur faible , valeur forte , valeur faible.

7.2 Apprentissage à états contraints (figure 1.13)

Nous avons développé un second type d'apprentissage : l'apprentissage à états contraints. En effet, l'apprentissage de type états supervisés impose que le superviseur choisisse le codage des états internes de la machine séquentielle connexionniste. Ce codage est plus ou moins

arbitraire et donc plus ou moins adapté au problème traité. Certains codages conviennent mieux que d'autres à l'apprentissage de diverses tâches séquentielles.

L'apprentissage à états contraints laisse quelques libertés au réseau neuronal dans le choix de la représentation des états internes. Le codage des états internes n'est pas complètement fixé par le superviseur. Le superviseur définit des intervalles de valeurs pour le codage des états internes. Nous présentons seulement les points modifiés par rapport au modèle précédent.

2' : La minimisation n'a pas besoin d'être réalisée par le superviseur. L'apprentissage à états contraints apporte une certaine souplesse au niveau du codage permettant de rassembler dans une même représentation les états dupliqués.

3' : Le superviseur impose des contraintes sur le codage des états internes. Ces contraintes apportent une certaine souplesse au niveau du codage permettant de réduire les possibilités de codages incompatibles.

5, 6 : L'algorithme de l'apprentissage à états contraints est identique à celui de l'apprentissage à états supervisés, si l'on prend soin de calculer l'erreur comme suit. Le codage par intervalle impose que lorsque la valeur de l'état du neurone appartient à l'intervalle, il n'y a pas d'erreur et par conséquent pas de modifications des poids du réseau. Par contre, si la valeur d'un neurone est hors de l'intervalle, une erreur est générée. L'erreur est la différence entre le milieu de l'intervalle désiré et la valeur calculée pour la cellule. Dans le cas ou l'intervalle est de la forme [-1, 1], il n'y a jamais d'erreur générée au cours de l'apprentissage.

<u>Conclusion</u>

L'apprentissage à états contraints apporte une certaine souplesse au niveau du codage permettant :

- de réduire les possibilités de codages incompatibles,
- de rassembler dans une même représentation les états dupliqués,
- de tracer le passé en multipliant les configurations d'états internes. En effet, l'apprentissage à états contraints multiplie les configurations d'états internes. Il est parfaitement possible, en utilisant cette redondance, de retracer l'histoire du réseau. Nous pouvons reconstruire la séquence des éléments d'entrée d'après la séquence de sortie obtenue. Ceci n'est pas possible dans le cas de l'étude des sorties de la fonction de transition d'états de l'automate détecteur équivalent.

7.3 Apprentissage à états non supervisés (figure 1.14)

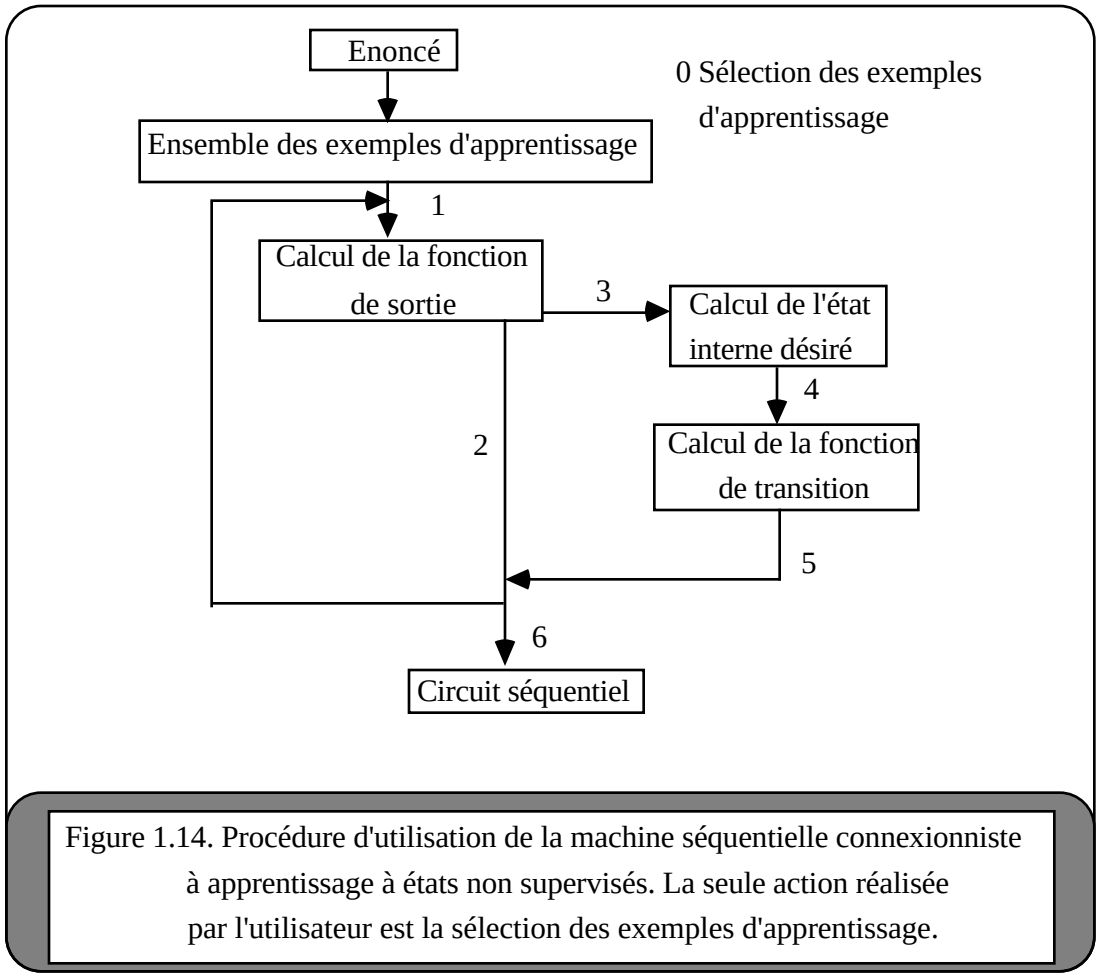

Figure 1.14. Procédure d'utilisation de la machine séquentielle connexionniste à apprentissage à états non supervisés. La seule action réalisée par l'utilisateur est la sélection des exemples d'apprentissage.

1 : Phase de propagation dans le modèle, on obtient ainsi un état interne et une valeur de sortie. Modification éventuelle de la fonction de sortie pour obtenir la sortie désirée à partir de l'état interne calculé. L'algorithme d'apprentissage est la rétro-propagation de gradient. Si malgré les modifications de poids l'apprentissage n'est pas possible, alors il faut modifier la fonction de transition. Ce sont les actions 3, 4.

2 : La modification de la fonction de transition a suffit pour obtenir la sortie désirée. On passe à l'exemple d'apprentissage suivant.

3 : Il faut modifier la fonction de transition. Dans une première étape, un algorithme de rétropropagation de la valeur désirée est utilisé pour déterminer l'état interne désiré. L'état interne désiré est l'état qui permet d'obtenir la sortie correcte sans modification de la fonction de sortie.

4 : Utilisant la valeur désirée obtenue en 3, l'algorithme de la rétro-propagation est appliqué sur la fonction de transition.

5 : Eventuellement, passage à l'exemple suivant.

6 : La réalisation du circuit séquentiel est terminée. La machine connexionniste montre un comportement correspondant aux exemples appris.

Conclusion

Ce modèle montre les propriétés suivantes :

- Acquisition du comportement d'un automate sans connaissance a priori du graphe des états.
- Détermination a priori de la nature séquentielle ou combinatoire du problème soumis.

8/ Les différentes approches des réseaux de neurones séquentiels dans le formalisme de la machine séquentielle connexionniste

Dans cette partie, nous représentons deux approches des réseaux de neurones séquentiels avec le formalisme utilisé pour décrire la machine séquentielle connexionniste. Les notions de fonction de transition et de fonction de sortie y sont explicites. Nous verrons comment sont réalisées ces fonctions pour chacune des approches et nous pourrons présenter les fonctions réalisables par ces modèles.

8.1 Modèle de Jordan

Le modèle de Jordan (figure 1.15) est très proche d'une machine séquentielle connexionniste dont la fonction de transition se réduit à une seule couche de neurones et la fonction de sortie à un réseau multi-couches (3 couches). On ajoute au modèle classique de la machine séquentielle connexionniste une boucle depuis la sortie sur l'entrée secondaire de la fonction de transition. De plus, il n'y a pas d'entrée primaire sur la fonction de transition.

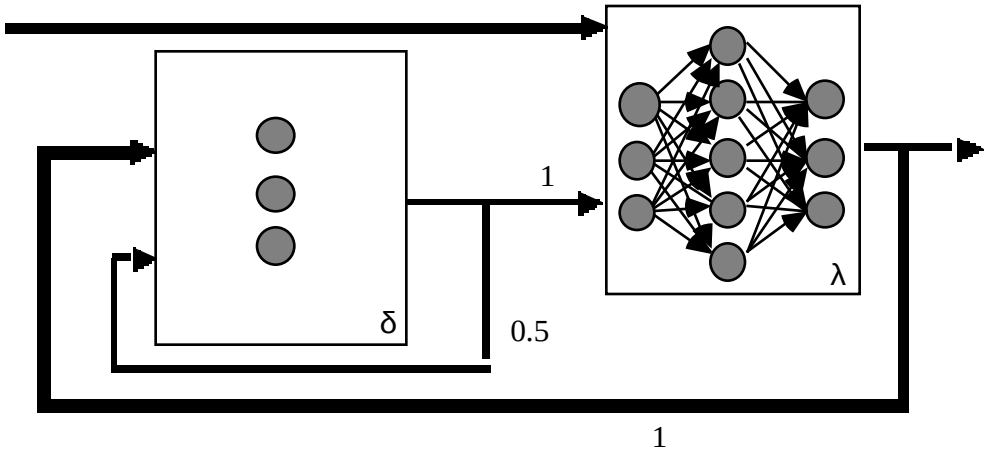

Figure 1.15. Présentation du modèle de Jordan avec le formalisme de la machine séquentielle connexionniste. On a indiqué à côté des connexions fixes leur poids.

La fonction de transition est fixée : les connexions sont de poids fixes égaux à 1 pour les connexions en provenance de la fonction de sortie et égaux à 0.5 pour les connexions depuis la fonction de transition.

La fonction de transition est apprise : les poids des connexions sont modifiés durant la phase d'apprentissage par l'algorithme de la rétropropagattion de gradient.

Pas de représentation explicite de l'état interne. La représentation utilisée est celle de la sortie. A ce sujet, l'utilisation d'intervalles pour coder la sortie autorise, dans une certaine mesure, le réseau à développer une représentation interne utile. Mais la fonction réalisée reste fixée.

Il est impossible de séparer l'état interne de la sortie. Ce modèle peut apprendre un graphe d'état, mais ne peut pas associer des sorties différentes à un même état. Il se comporte comme une machine d'état.

Le modèle de Jordan se ramène à une machine séquentielle connexionniste dont la fonction de transition est fixée. Cette fonction est plus simple, elle ne compte qu'une seule couche de poids.

8.2 Le modèle d'Elman

Le modèle d'Elman (figure 1.16) correspond à une machine séquentielle connexionniste dont les fonctions de transition et de sortie sont réduites à des réseaux de neurones à deux couches. Il n'y a pas d'entrée primaire sur la fonction de sortie.

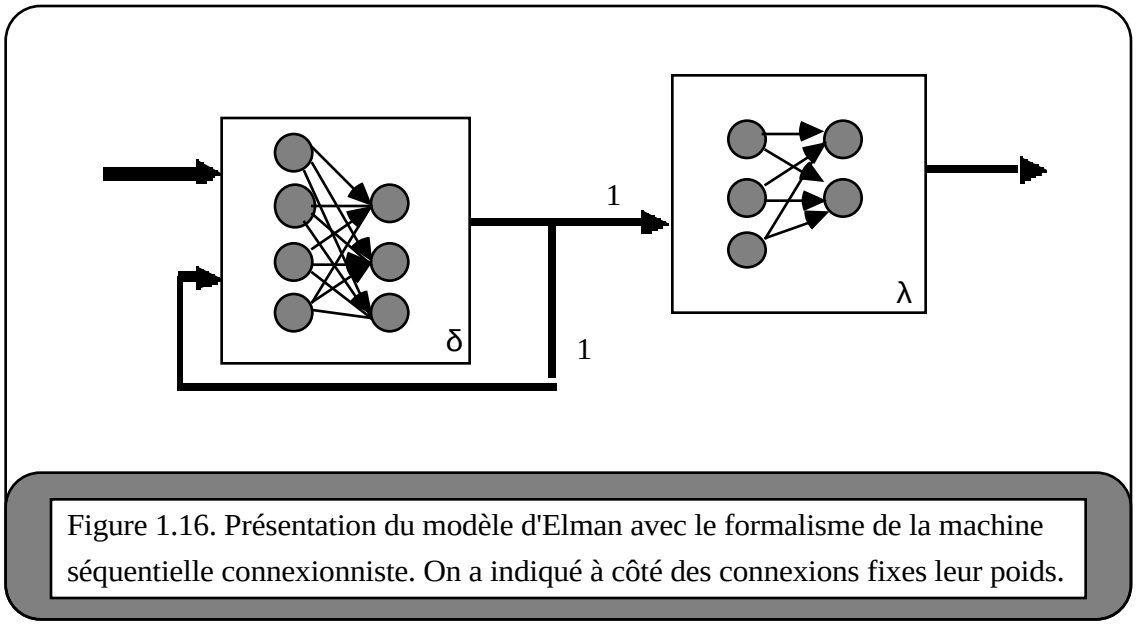

Figure 1.16. Présentation du modèle d'Elman avec le formalisme de la machine séquentielle connexionniste. On a indiqué à côté des connexions fixes leur poids.

La fonction de transition et la fonction de sortie sont apprises. Cependant, il n'y a qu'une seule couche de poids modifiables pour chaque fonction. Ceci limite les possibilités au niveau des fonctions réalisables. Seules peuvent être réalisées des fonctions linéairement séparables.

L'apprentissage modifie à la fois la fonction de transition et la fonction de sortie. Ainsi, si la sortie est fausse, mais que l'état interne est bon, on va néanmoins modifier les deux fonctions et inversement. L'algorithme est la rétropropagation de gradient.

Du fait de sa décomposition en fonction de transition et fonction de sortie, ce modèle est capable d'apprendre à se comporter comme un automate à états finis. Cependant, ses possibilités sont limitées par le fait que chacun des réseaux qui compose les fonctions ne comporte qu'une seule couche de poids et qu'il y a modification de la fonction de transition et de la fonction de sortie durant l'apprentissage.

Dans le modèle d'Elman, à la fois la fonction de transition et la fonction de sortie sont modifiées par une itération d'apprentissage. Cependant, ce modèle développe le concept d'état interne, et il sait le représenter. C'est une différence par rapport à la machine séquentielle connexionniste où le codage de l'état interne n'est pas choisi par le réseau. Les possibilités des fonctions de transition et de sortie sont limitées car il n'y a qu'une seule couche de poids modifiables pour chacune. Rappelons à ce sujet que la machine séquentielle connexionniste est plus performante, chaque fonction étant représentée par un réseau multicouches.

8.3 Conclusion

Le modèle de Jordan est une machine séquentielle connexionniste dont la fonction d'état est fixée. Le modèle proposé par Elman correspond à une machine séquentielle connexionniste dont les possibilitées des fonctions de sortie et d'état sont limitées. Il est possible de décrire dans ces termes tous les autres modèles de réseaux séquentiels. Le modèle de la machine séquentielle connexionniste se présente comme une généralisation des modèles de réseaux de neurones séquentiels. Chaque fonction y est réalisée par un réseau de neurones multicouches.

A partir des fonctions réalisables par chaque modèle, il est possible d'appréhender les possibilités face à un domaine d'application. Ainsi, le modèle de Jordan est une machine d'état. Le modèle d'Elman est une machine séquentielle dont les fonctions de sortie et d'état sont linéaires. Le modèle de la machine séquentielle connexionniste est le plus puissant, chaque fonction pouvant être non linéaire.

CHAPITRE II

UNE MACHINE SEQUENTIELLE CONNEXIONNISTE

Dans ce chapitre, nous détaillons un modèle de la machine séquentielle connexionniste [Touzet 89a], [Touzet 90a]. La première partie aborde la structure. La seconde partie explicite le comportement du modèle. Enfin, la dernière partie traite de trois apprentissages développés en s'appuyant sur des exemples illustratifs. Les résultats obtenus sur des problèmes de reconnaissance de séquences sont présentés.

1 Structure

___Nous reprenons ici plus en détails la structure de notre modèle, déjà présenté succintement au chapitre précédent. Ainsi que le rappelle la figure 2.1, il comporte deux parties :

- l'une réalise la fonction de transition,
- l'autre réalise la fonction de sortie.

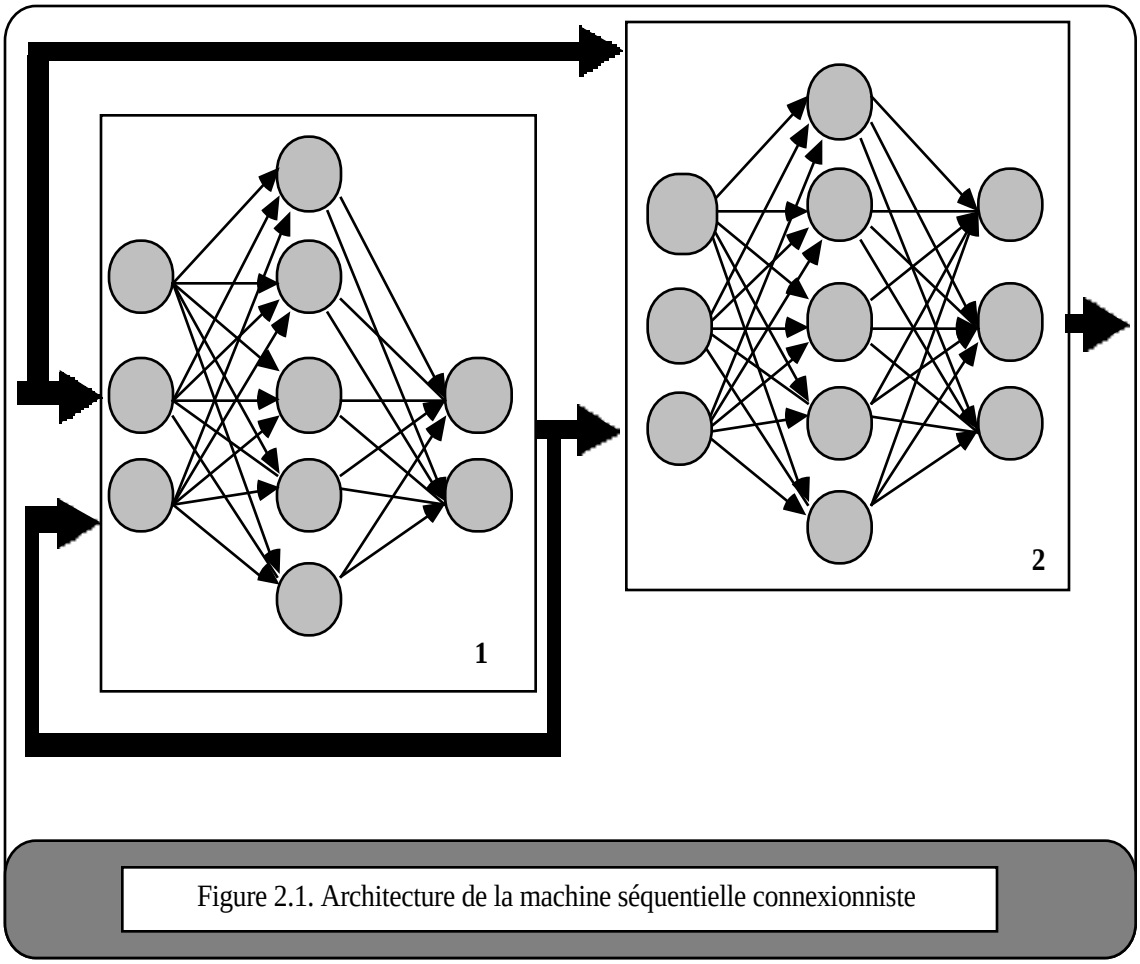

Figure 2.1. Architecture de la machine séquentielle connexionniste

Chacune de ces deux fonctions est représentée par un bloc. Le bloc 1 réalise la fonction de transition et le bloc 2 la fonction de sortie. Chaque bloc est un réseau de neurones multi-couches. Le bloc 1 reçoit les entrées primaires, ainsi que des boucles de rétro-actions depuis sa sortie. Le bloc 2 reçoit lui aussi les entrées primaires plus la sortie du bloc 1.

2.1 Description neuronale

La figure 2.2 détaille l'architecture neuronale de la machine séquentielle connexionniste. Les neurones qui composent cette structure sont tous identiques. Par contre, les connexions se répartissent en deux classes.

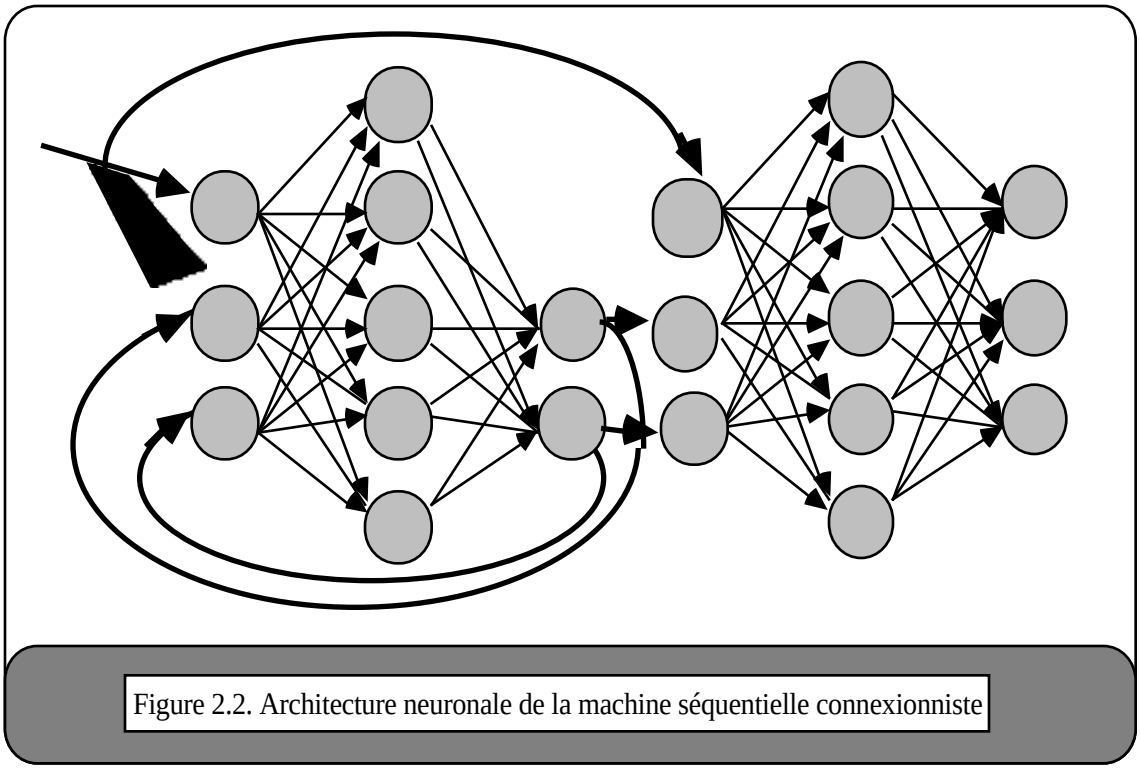

Figure 2.2. Architecture neuronale de la machine séquentielle connexionniste

Les connexions en provenance des entrées extérieures et celles en provenance de la sortie du réseau de transition sont de poids fixes. Ces poids ne sont pas modifiés lors de l'apprentissage. Les connexions récurrentes sont univoques : chaque neurone ne possède qu'un seul successeur et réciproquement.

Les connexions entre les différentes couches dans chaque réseau sont plastiques, leur poids est modifié durant l'apprentissage. Tous les neurones d'une couche sont connectés à tous les neurones de la couche suivante. Il n'y a pas de connexions entre neurones d'une même couche, ainsi que de connexion récurrente sur le neurone lui-même.

2.2 Dénominations

Les neurones se répartissent en couches, il y a trois types de couches (figure 2.3) :

- Couche d'entrée,
- Couche cachée,
- Couche de sortie.

Nous désignons sous le terme neurones d'entrées primaires les neurones d'entrée directement connectés à une entrée extérieure et sous le terme neurones d'entrées secondaires les neurones d'entrée non directement connectés à une entrée extérieure.

Nous employons abusivement le terme réseau de transition (respectivement réseau de sortie) pour désigner le réseau de neurones réalisant la fonction de transition (respectivement le réseau de neurones réalisant la fonction de sortie). Les neurones du réseau réalisant la fonction de transition (respectivement les neurones du réseau réalisant la fonction de sortie) sont appelés neurones de la fonction de transition (respectivement neurones de la fonction de sortie).

Réseau de transition : La couche d'entrée se compose de neurones d'entrées primaires en relation directe avec l'extérieur et de neurones d'entrées secondaires connectés à la sortie du réseau de transition. Ces neurones d'entrées secondaires codent l'état présent, nous les appelerons neurones d'état présent. Les couches cachées des blocs fonction de transition et fonction de sortie sont semblables. Les neurones qui les composent sont dénommés neurones cachés. Les neurones de la couche de sortie sont appelés neurones de sortie de la fonction de transition.

Réseau de sortie : La couche d'entrée de la fonction de sortie est composée de neurones d'entrées secondaires connectés à la sortie du réseau de transition et de neurones d'entrées primaires. Les neurones de la couche de sortie codent la réponse de la machine, ils sont appelés neurones de sortie.

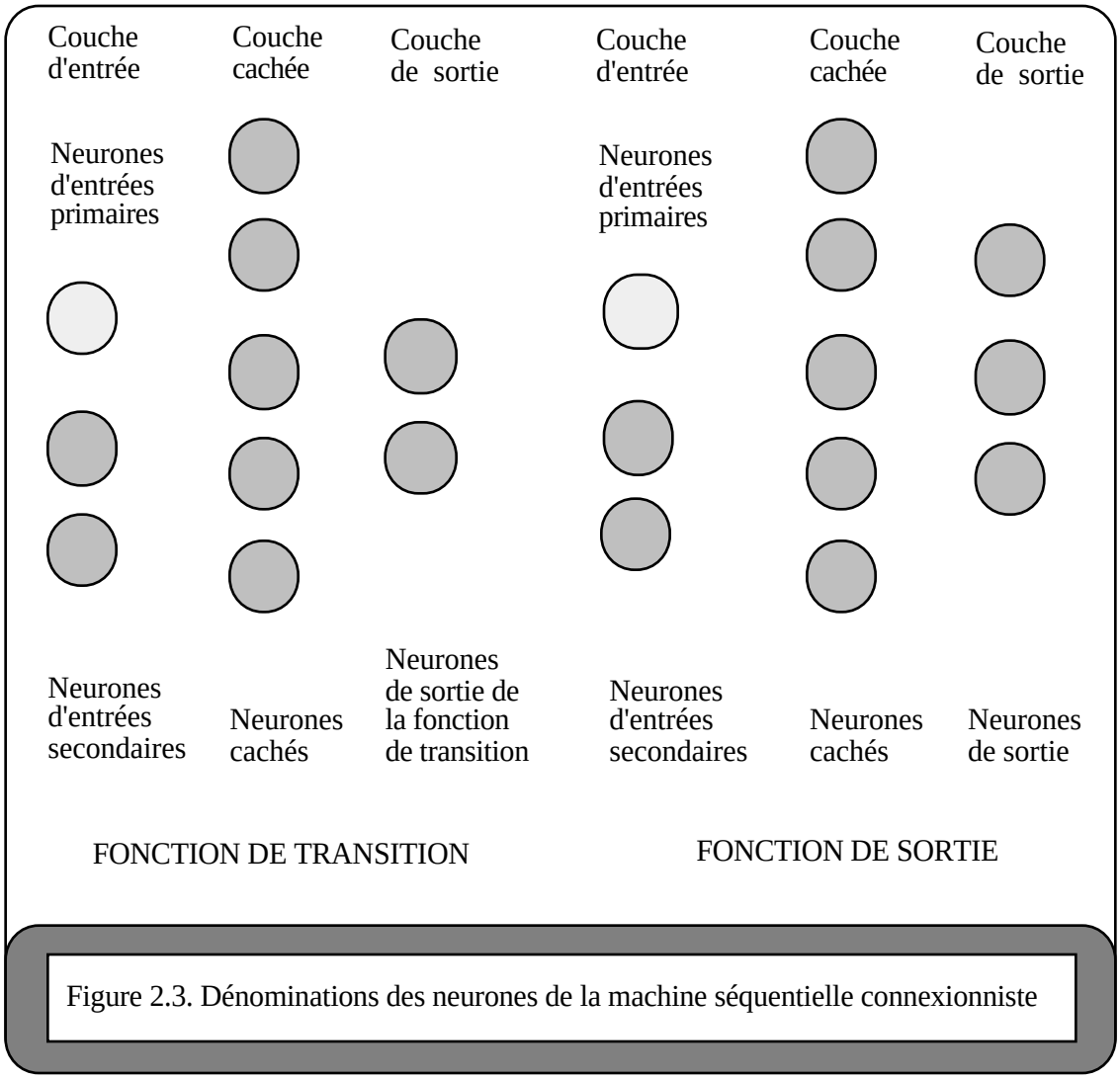

Figure 2.3. Dénominations des neurones de la machine séquentielle connexionniste

Connexions : La figure 2.4 donne le type des connexions entre les différentes catégories de neurones. Les connexions sont orientées. Les neurones de départ sont représentés sur les lignes du tableau. Seules les connexions existantes ont été indiquées. Celles notées **M** sont des connexions modifiables, leur poids est ajusté durant la phase d'apprentissage. Les connexions notées **F** sont de poids fixes.

Les connexions en provenance des entrées extérieures et celles en provenance de la sortie du réseau de transition sont de poids fixes.

Les connexions entre les différentes couches dans chaque réseau sont modifiables. Tous les neurones d'une couche sont connectés à tous les neurones de la couche suivante.

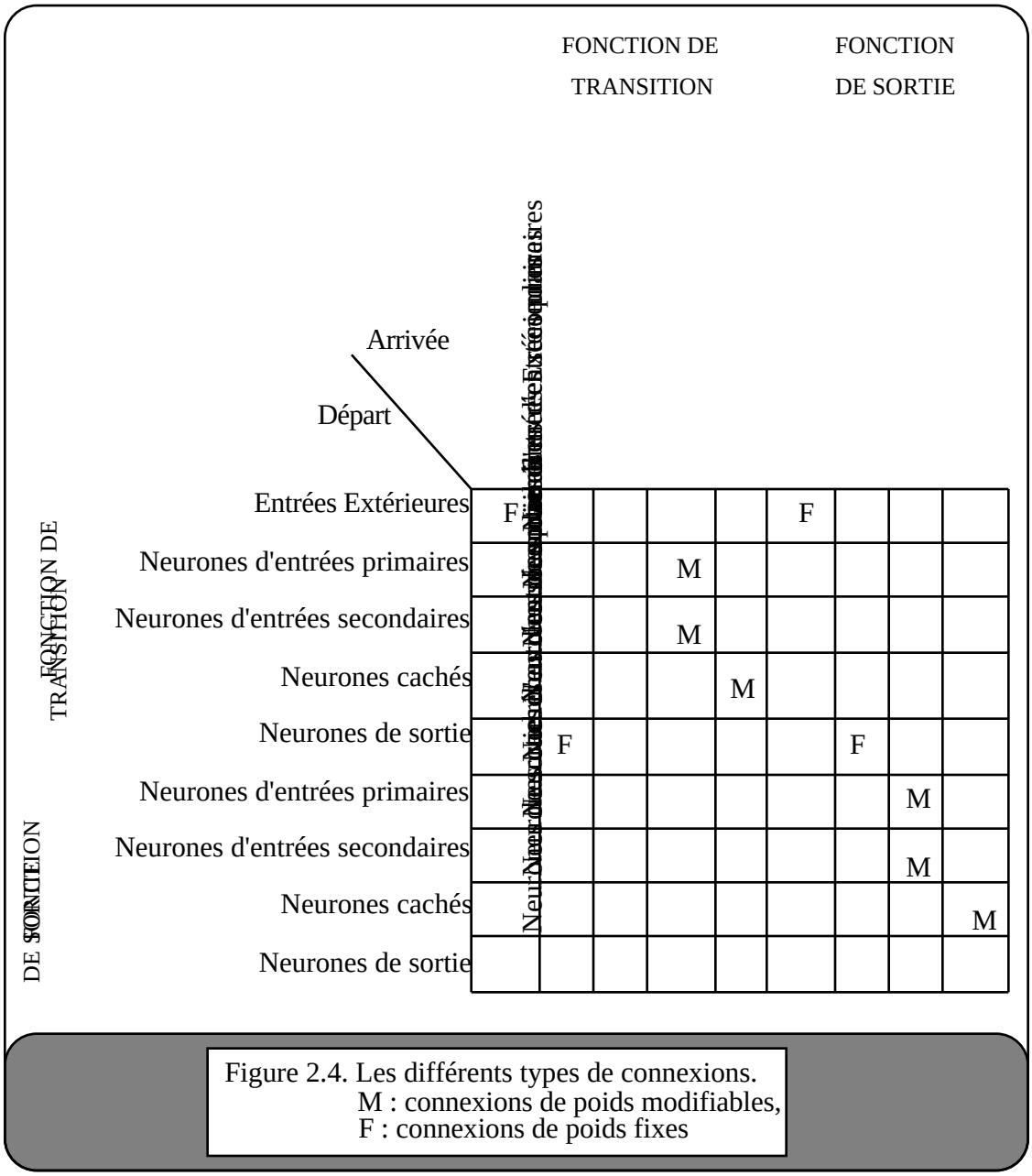

Figure 2.4. Les différents types de connexions.
M : connexions de poids modifiables,
F : connexions de poids fixes

Les deux types de connexions (fixes et modifiables) définissent une structure où l'on retrouve deux réseaux multicouches classiques (dotés de connexions modifiables) mis en relation par des connexions fixes. Chacun des réseaux multicouches correspond à un bloc fonctionnel de la machine séquentielle. C'est par rapport à leur fonction que nous avons désigné chacun des types de neurones (neurone d'entrée primaire, secondaire, …). Le fonctionnement du modèle est l'objet de la prochaine partie.

2 Fonctionnement

Dans cette partie, nous détaillons le fonctionnement de la machine séquentielle connexionniste. Nos expérimentations ont toutes trait à l'apprentissage et à la reconnaissance de séquences. Ceci nous amène à définir la forme d'une séquence. Le ou-exclusif séquentiel est utilisé comme exemple pédagogique. Il nous permet d'introduire des notions relatives au graphe d'états et d'état interne, qui sont au coeur de nos interprétations. Le paragraphe suivant illustre le comportement du modèle sur cet exemple. Nous détaillons, en particulier, l'étape du codage des formes d'entrées et des états internes sur le modèle, le comportement des neurones et l'influence des connexions (notamment celles de poids fixes).

2.1 Fonctionnement général

La figure 2.5 illustre le fonctionnement général. Le réseau de transition calcule à partir de l'état présent et de l'entrée extérieure l'état interne. Cet état interne est le prochain état présent. Le réseau de sortie calcule la sortie à partir de l'état présent et de la même entrée extérieure.

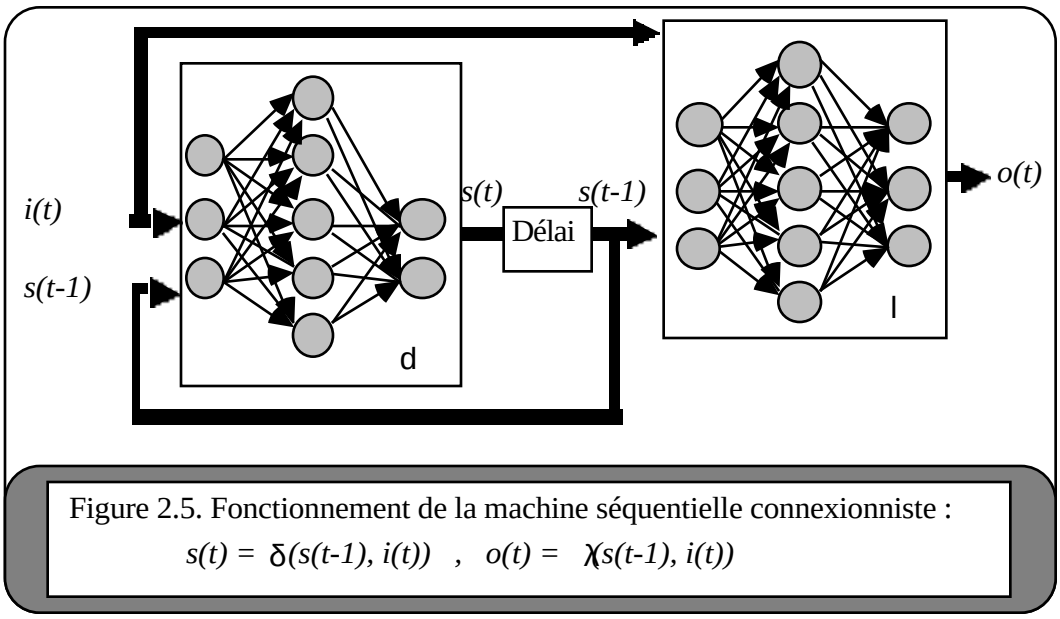

Figure 2.5. Fonctionnement de la machine séquentielle connexionniste :
$s(t) = \delta(s(t-1), i(t))$, $o(t) = \chi(s(t-1), i(t))$

Le comportement de ce modèle est celui d'une machine séquentielle déterministe synchrone. Toutes les modifications d'état des entrées primaires et secondaires ne sont prises en compte qu'à certaines dates précises, fixés par une horloge. En dehors de ces instants particuliers, le modèle est bloqué car la boucle de rétro-action est ouverte. Celle-ci ne se ferme qu'à chaque instant d'horloge et permet alors la remise à jour de l'état présent.

2.2 Notion de séquences

Pour la suite de notre étude, nous définissons une séquence comme une suite ordonnée de formes. La longueur de la séquence est le nombre de formes qui la compose. Une forme est le classique vecteur d'entrée des réseaux de neurones combinatoires.

Dans une séquence, l'ordre de présentation des formes est primordiale. Intervertir deux formes d'une séquence modifie totalement le résultat. En fait, une nouvelle séquence est créée. Dans une séquence, la même forme d'entrée peut produire des sorties différentes. Inversement, deux formes d'entrée différentes peuvent produire la même sortie.

Comment lire la séquence ? 0 1 -> 0 1

On trouve à gauche de la flèche les formes présentées en entrée du système (aussi appelées formes d'entrée). La forme la plus à gauche est présentée la première. La forme suivante à être présentée se trouve à sa droite immédiate et ainsi de suite. Les réponses attendues du système sont à droite de la flèche. Elles sont aussi appelées valeurs de sortie associées. L'ordre chronologique des réponses est représenté de la gauche vers la droite. Cette séquence se lit : 0 est présenté au sytème, la réponse est 0. Puis 1 est présenté au système dont la réponse attendue est 1.

Le ou-exclusif séquentiel est un petit problème séquentiel qui nous servira à illustrer le fonctionnement et l'apprentissage. Il est décrit par les quatre séquences suivantes :

- 0 1 -> 0 1(i)
- 1 0 -> 0 1 (ii)
- 0 0 -> 0 0 (iii)
- 1 1 -> 0 0 (iiii)

La figure 2.6 montre le graphe d'état de l'automate réalisant le ou-exclusif séquentiel. Les états internes de l'automate sont indiqués par des cercles. Les transitions entre les états sont représentées par des arcs orientés. Sur les arcs de transition est noté le couple : Forme d'entrée/Valeur de sortie associée.

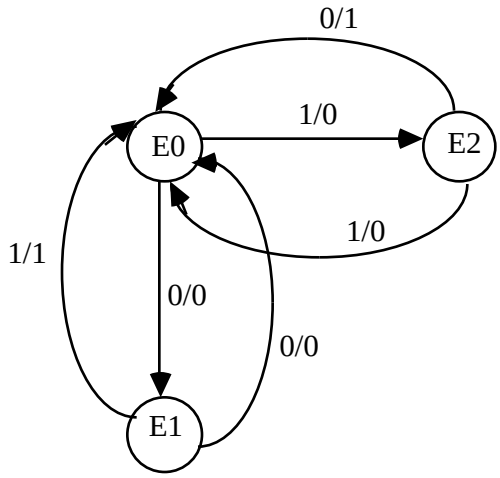

Figure 2.6. Graphe d'état d'un automate réalisant le ou-exclusif séquentiel.

La figure 2.6 fait apparaître les états internes de l'automate. Ces états internes notent l'historique. Il y a deux histoires possibles pour cet automate selon que la forme précédemment présentée était 1 ou 0. A l'origine, l'automate est dans un état initial, noté E0. Une première forme est présentée. Si c'est un 0 (respectivement 1), le système se place dans l'état interne noté E1 (respectivement E2). La sortie de l'automate est 0 quelle que soit la valeur de l'entrée. Une seconde forme est présentée. Dans tous les cas, le système revient dans l'état initial E0. La sortie est égale à 1 (respectivement 0) si la seconde forme est 1, elle est égale à 0 (respectivement 1) si la seconde forme est 0.

Nous pouvons compléter les équations (i) ... (iiii) de façon à faire apparaitre les états internes :

-	(0, E0)(1, E1) ->	(E1, 0)	(E0, 1)	(i)'
-	(1, E0)(0, E2) ->	(E2, 0)	(E0, 1)	(ii)'
-	(0, E0)(0, E1) ->	(E1, 0)	(E0, 0)	(iii)'
-	(1, E0)(1, E2) ->	(E2, 0)	(E0, 0)	(iiii)'

La première équation (i)' se lit : 0 est présenté, le système est dans l'état E0. Le système évolue dans un état E1, la réponse est 0. Puis, 1 est présenté au système dans l'état E1. Il évolue dans l'état E0, la réponse attendue est 1. Les équations suivantes se lisent de la même manière.

2.3 Illustration : le ou-exclusif séquentiel

La figure 2.7 illustre le fonctionnement déterministe synchrone du modèle.

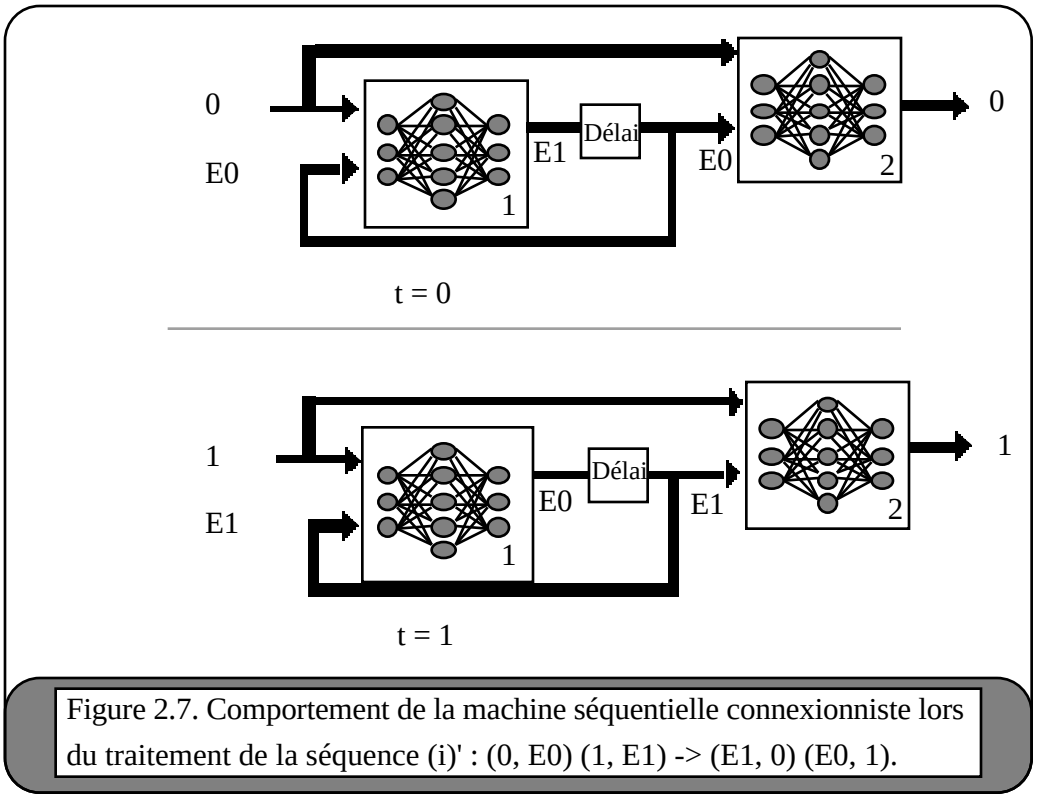

Figure 2.7. Comportement de la machine séquentielle connexionniste lors du traitement de la séquence (i)' : (0, E0) (1, E1) -> (E1, 0) (E0, 1).

A la date t = 0, la machine est dans l'état initial (E0). E0 est appliqué sur les entrées secondaires. La forme d'entrée (0) est appliquée sur les entrées primaires. Par propagation le bloc 1 calcule le prochain état présent (E1) et le bloc 2 donne la valeur de la sortie (0).

A la date t = 1, il y a remise à jour des entrées secondaires : l'état présent est E1. D'autre part, la forme d'entrée (1) est appliquée sur les entrées primaires. Par propagation le bloc 1 calcule le prochain état présent (E0) et le bloc 2 donne la valeur de la sortie (1).

Le codage des formes d'entrée et des états internes est le suivant :

Les entrées extérieures : un seul neurone d'entrée primaire code la valeur de l'entrée. Celui-ci vaut 1 (réciproquement 0) lorsque l'entrée vaut 1 (réciproquement 0).

Les états internes : E0, E1 et E2 sont codés sur deux neurones d'entrées secondaires. Le codage est le suivant :

- E0 -> 0 - E1 -> 0 - E2 -> 1
  0 1 0

En fonction du codage adopté, les équations (i)' ... (iiii)' se re-écrivent :

- 0 1 -> 0 1 (i)''
  0 0 0 0
  0 1 1 0

- 1 0 -> 0 1 (ii)''
  0 1 1 0
  0 0 0 0

- 0 0 -> 0 0 (iii)''
  0 0 0 0
  0 1 1 0

- 1 1 -> 0 0 (iiii)''
  0 1 1 0
  0 0 0 0

Sur la ligne supérieure, on trouve à gauche, les valeurs à appliquer sur les entrées primaires, et à droite les valeurs de sortie désirées. Sur les deux lignes inférieures sont indiqués à gauche les valeurs à appliquer sur les entrées secondaires et à droite le codage du prochain état présent.

La figure 2.8 illustre le comportement des neurones de la machine séquentielle synchrone lors du traitement de la séquence (i)". Elle fait suite à la figure 2.7 qui décrit le fonctionnement de la machine au niveau des blocs fonctionnels.

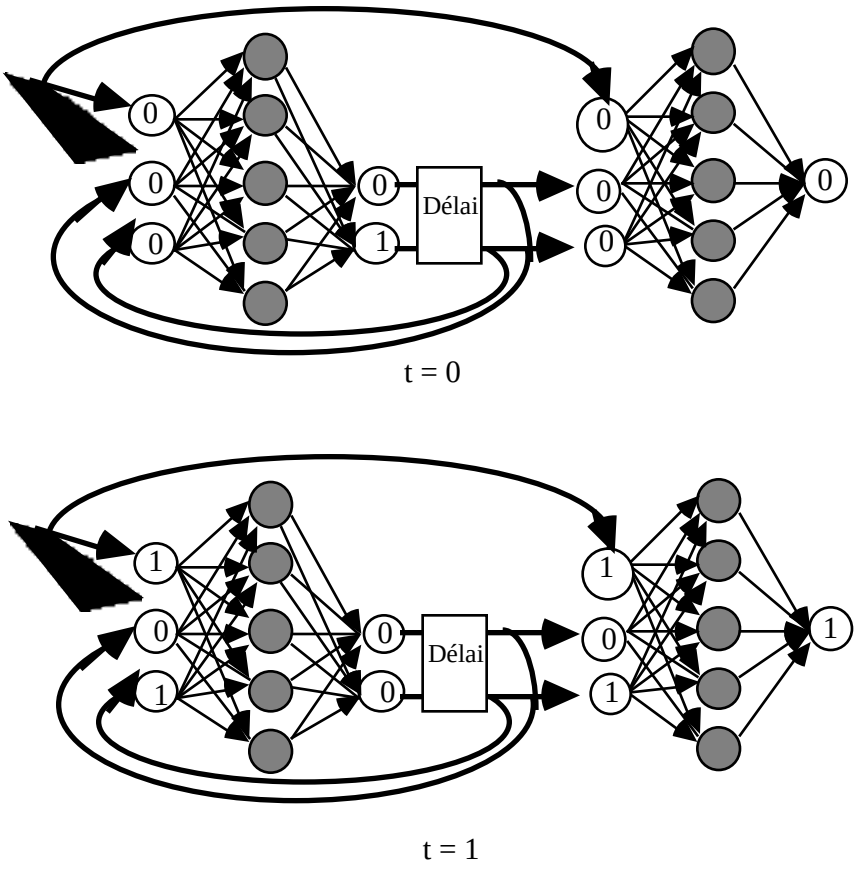

Figure 2.8. Comportement des neurones de la machine séquentielle connexionniste lors du traitement de la séquence (i)" :

0	1	->	0	1
0	0		0	0
0	1		1	0

A la date t = 0 : les connexions récurrentes étant ouvertes, sur les neurones d'entrées primaires est appliquée la valeur (0) et sur les neurones d'entrées secondaires est appliqué l'état présent (0, 0). Par propagation, les valeurs des neurones de sortie du réseau de transition sont (0, 1) et la valeur du neurone de sortie est 0.

A la date t = 1 : la valeur 1 est appliquée sur les neurones d'entrées primaires. Simultanément, la fermeture des connexions récurrentes est réalisée. Les valeurs des neurones d'état présent sont ainsi mises à jour (0, 1). Les connexions sont ouvertes à nouveau. Par propagation, les valeurs des neurones de sortie du réseau de transition sont (0, 0) et la valeur du neurone de sortie est 1.

Le comportement des neurones composant la machine séquentielle connexionniste est décrit par les équations suivantes :

- $x_i = f(a_i) = (e^{\partial a_i - 1}) / (e^{\partial a_i + 1})$ (1)

- $a_i = \sum_j w_{ij} \cdot x_j$ (2)

x_i est la valeur de l'état du neurone i. f est la fonction de transfert du neurone. C'est une fonction continue de type sigmoïde dont le comportement est représenté figure 2.9. a_i est la somme pondérée des entrées sur le neurone i , w_{ij} désigne le poids d'une connexion du neurone i vers le neurone j, x_j est la valeur de l'état du neurone j. ∂ est une constante (couramment égale à 0.2).

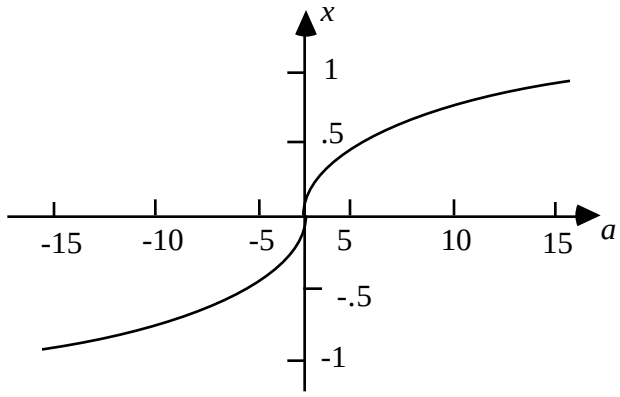

Figure 2.9. Comportement de la fonction de transfert du neurone correspondant à l'équation (1).

Dans ce modèle, tout neurone qui reçoit une connexion de poids fixe possède un prédecesseur unique. Lorsque le neurone j est connecté au neurone i par une connexion de poids fixe, tout se passe comme-ci le neurone i transcrivait la valeur de l'état du neurone j. Ceci est illustré sur la figure 2.10.

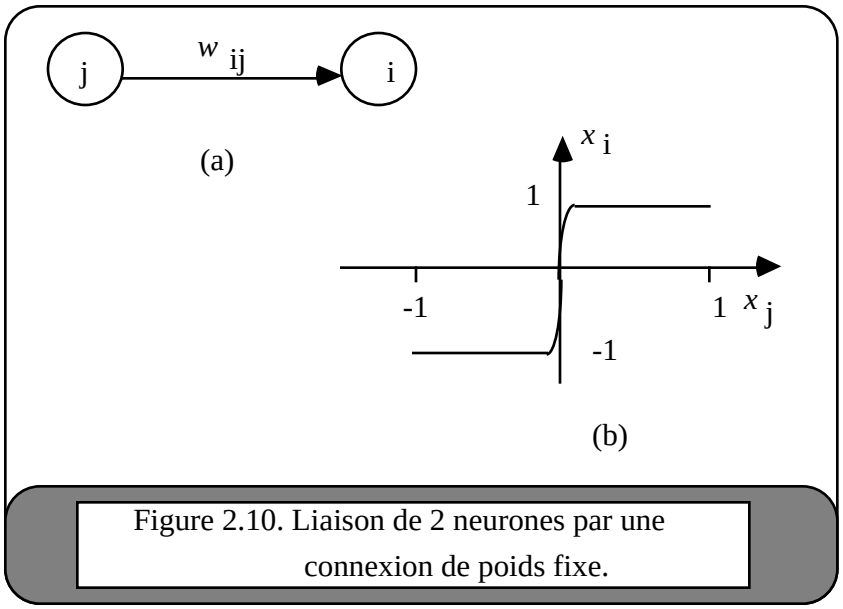

Figure 2.10. Liaison de 2 neurones par une connexion de poids fixe.

Sur la figure 2.10.a, le neurone j est connecté au neurone i. Cette connexion est de poids fixe. Ce sera donc la seule connexion à destination du neurone i.

La figure 2.10.b montre la valeur de l'état du neurone i (x_i) en fonction de l'état du neurone j (x_j). La connexion de poids fixe a pour fonction de recopier dans le neurone successeur, la valeur de l'état du neurone prédecesseur. Cependant cette recopie n'est pas toujours la fonction identité (poids égaux à 1). La valeur transmise est modifiée de façon à obtenir pour la valeur de l'état du neurone successeur, des valeurs proches de -1 et +1 (par exemple : poids égaux à 15). Ceci est particulièrement intéressant dans le cadre d'un codage en (-1, +1) sur le réseau, car il est ainsi possible de réduire une dérive éventuelle.

En conclusion, le comportement est celui d'une machine séquentielle déterministe synchrone. Nous avons illustré la phase de propagation avec l'exemple pédagogique du ou-exclusif séquentiel. Chacune des étapes nécessaires à l'exécution a été explicité : définition des séquences, codage des entrées et des états internes. Il est à noter que le traitement réalisé par les connexions de poids fixes semble prometteur, bien que encore peu utilisé et largement dépendant du codage choisi.

3 Apprentissage

L'apprentissage sur le modèle de la machine séquentielle connexionniste fait appel à l'algorithme classique de la rétropropagation de gradient. Nous avons étudié différents types d'apprentissage : apprentissage de type à états supervisés, à états contraints et non supervisés. Ces algorithmes d'apprentissage diffèrent par la connaissance qu'il faut avoir a priori du graphe des états. Nous illustrons les propriétés d'apprentissage sur un petit problème simple de détection de séquence de longueur 3.

3.1 Rappel sur l'algorithme d'apprentissage de la rétropropagation de gradient [Lippmann 87]

L'algorithme de la rétro-propagation est un algorithme de gradient itératif conçu pour minimiser un critère quadratique d'erreur entre la sortie obtenue d'un réseau multicouches et la sortie désirée. Cette minimisation est réalisée par une configuration des poids adéquate. L'erreur (ε) est la différence entre la valeur désirée (d) pour la cellule de sortie et sa valeur calculée par propagation (x). Ce signal d'erreur permet de définir une fonction de coût :

$$C(W) = M[C_i(W)] = M[\sum_j \varepsilon_{ij}^2(W)] \text{ avec } \varepsilon_{ij} = (d_{ij} - x_{ij})$$

où, j indique un numéro d'indice pour les cellules de sortie et i indique un exemple d'apprentissage. M est l'opérateur de moyennage, c'est une estimation de la moyenne temporelle dans le cas stochastique.

Cet algorithme nécessite une fonction continue, non-linéaire et différentiable comme fonction de transfert du neurone. Dans notre étude, nous employons la fonction de type sigmoïde (f) suivante :

$$f(a_i) = (e^{\partial a_i} - 1) / (e^{\partial a_i} + 1)$$

Etape 1 : Initialisation des poids à de petites valeurs aléatoires.

Etape 2 : Présentation de la forme d'entrée sur la couche d'entrée du réseau.

Etape 3 : Calcul de la sortie obtenue, par propagation.

Etape 4 : Modification des poids :

On utilise un algorithme récursif partant des neurones de sortie jusqu'aux neurones d'entrées. Les poids sont ajustés par l'application d'une procédure de gradient :

$$w_{ij}(t+1) = w_{ij}(t) + l \, . y_i \, . x_j$$

Dans cette équation, l est un gain fixé par l'utilisateur, y_i est un terme d'erreur pour le neurone i. Si le neurone i est une cellule de sortie alors :

$$y_i = 2 \, f'(a_i).(d_i - x_i)$$

Si le neurone i est une cellule cachée alors :

$y_i = f'(a_i) . \sum_k (w_{ki} . y_k)$

où k est un indice qui parcours tous les neurones des couches supérieures à i.

On appelle couche supérieure une couche comprise entre la couche actuelle et la couche de sortie.

Etape 5 : Retour à l'étape 2.

3.2 Apprentissage de type états supervisés

Le terme d'apprentissage supervisé est employé car l'utilisateur a à charge de spécifier complètement le codage des états internes. Dans ce cas, la valeur de sortie désirée du réseau de transition est connue, et l'apprentissage est réalisé par application de deux procédures de rétropropagation de gradient. Une pour le réseau de transition et une pour le réseau de sortie. Il y a une itération d'apprentissage après chaque présentation d'une forme de la séquence et non pas seulement à la fin de la séquence. Afin d'appliquer l'algorithme, il est nécessaire de définir un signal d'erreur pour chaque bloc. Les résultats sur un problème de détection de séquences de longueur 3 montrent que le système généralise un comportement appris avec des séquences de valeurs binaires à des séquences de valeurs discrètes.

Apprentissage du réseau de transition

Le codage des états internes fournit les données nécessaires à l'établissement d'un signal d'erreur sur la couche de sortie du bloc transition d'état. Ce codage est choisi par le superviseur. Il justifie le terme d'apprentissage à états supervisés.

Dans le cas du ou-exclusif séquentiel, le codage des états internes (sur deux neurones) est le suivant :

- E0 -> 0 - E1 -> 0 - E2 -> 1
  0 1 0

Les exemples d'apprentissage dans le cas du ou-exclusif séquentiel (4 séquences de longueur 2) sont décrits ci-dessous :

$\{[(0, E0) \rightarrow E1]_1,$
$[(1, E1) \rightarrow E0]_2\}_1$
$\{[(1, E0) \rightarrow E2]_1,$
$[(0, E2) \rightarrow E0]_2\}_2$
$\{[(0, E0) \rightarrow E1]_1,$

$[(0, E1) \rightarrow E0]_2\}_3$
$\{[(1, E0) \rightarrow E2]_1,$
$[(1, E2) \rightarrow E0]_2\}_4$

Apprentissage du réseau de sortie

Le signal d'erreur est la différence entre la valeur désirée et la valeur calculée par propagation pour chaque neurone de sortie. Les exemples d'apprentissage dans le cas du ou-exclusif séquentiel (4 séquences de longueur 2) sont les suivants :

$\{[(0, E0) \rightarrow 0]_1,$
$[(1, E1) \rightarrow 1]_2\}_1$
$\{[(1, E0) \rightarrow 0]_1,$
$[(0, E2) \rightarrow 1]_2\}_2$
$\{[(0, E0) \rightarrow 0]_1,$
$[(0, E1) \rightarrow 1]_2\}_3$
$\{[(1, E0) \rightarrow 0]_1,$
$[(1, E2) \rightarrow 0]_2\}_4$

Les exemples utilisés sont de la forme suivante :

$\{[(\text{Entrée, Etat présent}) \rightarrow (\text{Etat interne désiré, Sortie désirée})]_j\}_i$ pour la forme j de la séquence i.

Ci-dessous sont décrits les exemples d'apprentissage dans le cas du ou-exclusif séquentiel (4 séquences de longueur 2) :

$\{[(0, E0) \rightarrow (E1, 0)]_1,$
$[(1, E1) \rightarrow (E0, 1)]_2\}_1$
$\{[(1, E0) \rightarrow (E2, 0)]_1,$
$[(0, E2) \rightarrow (E0, 1)]_2\}_2$
$\{[(0, E0) \rightarrow (E1, 0)]_1,$
$[(0, E1) \rightarrow (E0, 0)]_2\}_3$
$\{[(1, E0) \rightarrow (E2, 0)]_1,$
$[(1, E2) \rightarrow (E0, 0)]_2\}_4$

3.2.1 Algorithme

Dans le cas stochastisque, l'algorithme d'apprentissage à états supervisés se décompose comme suit :

- Présentation d'une séquence d'apprentissage :
  - Pour chaque forme de cette séquence :
    - Présentation de la forme sur les cellules d'entrées primaires et présentation de l'état présent sur les cellules d'état présent. Calcul de l'état du réseau par propagation à l'aide des formules :

$$- x_i = f(a_i) = (e^{\partial a_i} - 1) / (e^{\partial a_i} + 1) \quad (1)$$

$$- a_i = \sum_j w_{ij} \cdot x_j \quad (2)$$

    - Pour le réseau réalisant la fonction de transition d'état : présentation de l'état interne désiré sur les cellules de sortie et calcul des gradients relatifs aux a_i par rétropropagation. Application d'une itération de la procédure de gradient.
    - Pour le réseau réalisant la fonction de sortie : présentation de la forme de sortie désirée sur les cellules de sortie et calcul des gradients relatifs aux a_i par rétropropagation. Application d'une itération de la procédure de gradient.

Il est important de noter qu'il y a deux impératifs pour réaliser l'apprentissage à états supervisés :

(1) Le superviseur doit connaître le graphe d'états de l'automate à synthétiser par la machine séquentielle connexionniste.

(2) Le superviseur doit choisir le codage des états internes pour la machine séquentielle connexionniste.

3.2.2 Exemples

Le ou-exclusif séquentiel

Soit une machine séquentielle connexionniste dont la fonction de transition d'état est constituée d'un réseau de 4 neurones sur la couche d'entrée, 3 neurones sur la couche cachée et 3 neurones de sortie et dont la fonction de sortie est constituée d'un réseau de 4 neurones sur la couche d'entrée, 2 neurones sur la couche cachée et 1 neurone de sortie.

Environ 500 itérations sont nécessaires pour que les exemples d'apprentissage soient parfaitement appris. Le critère utilisé pour définir parfaitement appris est que la valeur de

l'erreur doit être inférieure à 10^{-2}. C'est à dire $\varepsilon_{ij} = (d_{ij} - x_{ij}) < 10^{-2}$. La reconnaissance est parfaite quellequesoit la longueur de la séquence testée.

Un exemple de séquence de test est le suivant :
Entrée : 0101001011000110111010111
Sortie obtenue : 0101000100000101000101000

Le détecteur de la séquence de formes 010

La machine séquentielle connexionniste a été testée sur un petit problème de reconnaissance de séquence permettant d'examiner les capacités de généralisation. Il s'agit pour la machine séquentielle connexionniste de détecter la séquence de formes 010.

Base d'exemple : la figure 2.11 montre le graphe d'état de l'automate correspondant.

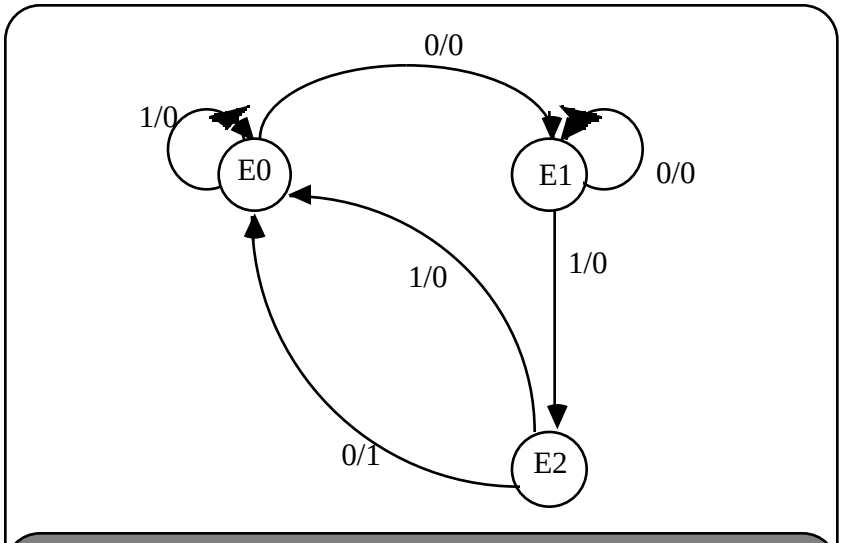

Figure 2.11. Graphe d'état de l'automate détecteur de la séquence de formes 010

Il y a 6 transitions possibles, qui contituent les exemples d'apprentissage :

$\{[(1, E0) \rightarrow (E0, 0)]_1\}_1$
$\{[(0, E0) \rightarrow (E1, 0)]_1\}_2$
$\{[(1, E1) \rightarrow (E2, 0)]_1\}_3$
$\{[(0, E1) \rightarrow (E1, 0)]_1\}_4$

$\{[(1, E2) \rightarrow (E0, 0)]_1\}_5$

$\{[(0, E2) \rightarrow (E0, 1)]_1\}_6$

Codage : Le codage des états internes est de la forme un parmi n (ici n=3). Les formes sont présentées à l'entrée de la machine sur 10 neurones. Le codage adopté est le suivant :

0 : -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

…

0.5 : -1 -1 -1 -1 -1 1 1 1 1 1

…

1 : 1 1 1 1 1 1 1 1 1 1

Codage de la sortie : la valeur 1 est codée par le neurone de sortie à +1 et la valeur 0 est codée par le neurone de sortie à -1.

Architecture : le réseau neuronal est constitué de 2 sous-réseaux. Celui réalisant la fonction de transition d'états comprend 10 neurones d'entrées primaires, 3 neurones d'état présent, 3 neurones d'état suivant, 8 neurones cachés pour la fonction de transition d'état. Le réseau réalisant la fonction de sortie est composé de 10 neurones d'entrée primaire, 3 neurones d'état interne, 4 neurones cachés et 1 neurone de sortie.

Apprentissage : 200 itérations d'apprentissage sont suffisantes pour que les séquences de formes binaires de longueurs arbitraires soient correctement traitées. Une sortie est considérée comme correcte lorsque la sortie obtenue est du même signe que la sortie désirée. C'est à dire : $(d_{ij}.x_{ij})>0$.

Cet exemple se représente aussi sous la forme suivante, figure 2.12 :

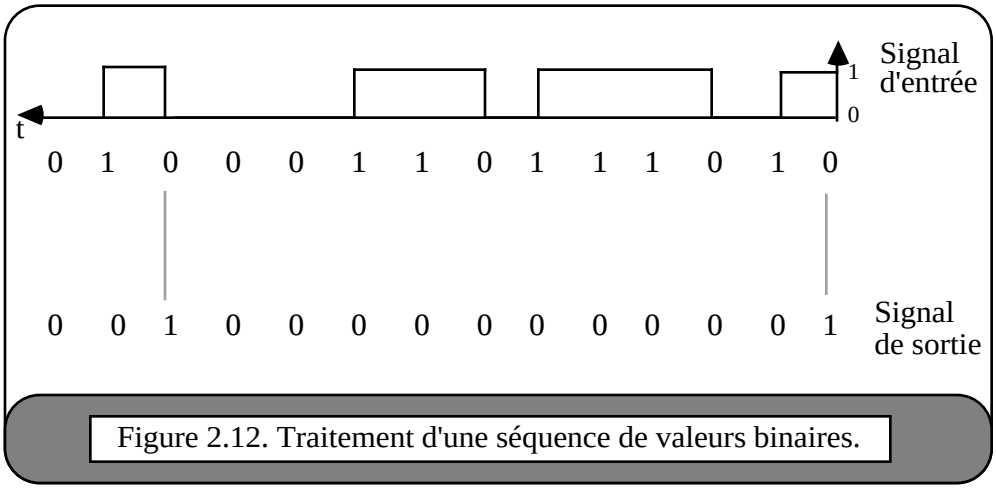

Figure 2.12. Traitement d'une séquence de valeurs binaires.

La ligne supérieure de la figure 2.12 montre la valeur du signal sur l'entrée du réseau en fonction du temps. Nous avons indiqué sous l'histogramme les valeurs exactes du signal. Il est à noter que la courbe montre une valeur continue dans le temps pour le signal d'entrée, alors que celui-ci est échantillonné. Chaque valeur d'échantillonnage est binaire. Sur la ligne inférieure est notée la valeur du signal de sortie rendue par le réseau, en correspondance avec l'entrée.

3.2.3 Généralisation

Après 200 itérations d'apprentissage sur les 6 transitions binaires, nous mesurons la généralisation. Dans ce cas, la possibilité de traiter correctement des séquences de formes autres que binaires, alors que l'apprentissage ne s'est déroulé qu'avec des séquences de formes binaires. Ainsi, dans le cas de la séquence suivante, qu'elle va être la réponse du réseau ? Est-il capable de reconnaitre la séquence 0.1 0.8 0.2 comme bonne ?

Entrée : 0.1 0.8 0.2 0.1 0.0 0.7 0.8 0.3 0.9 0.8 1.0 0.3 0.7 0.1
Sortie obtenue : 0 0 1 0 0 0 0 0 0 0 0 0 0 1
Rappelons que l'on considère correcte une sortie lorsque son signe est juste.

Cet exemple se représente aussi sous la forme suivante, figure 2.13 :

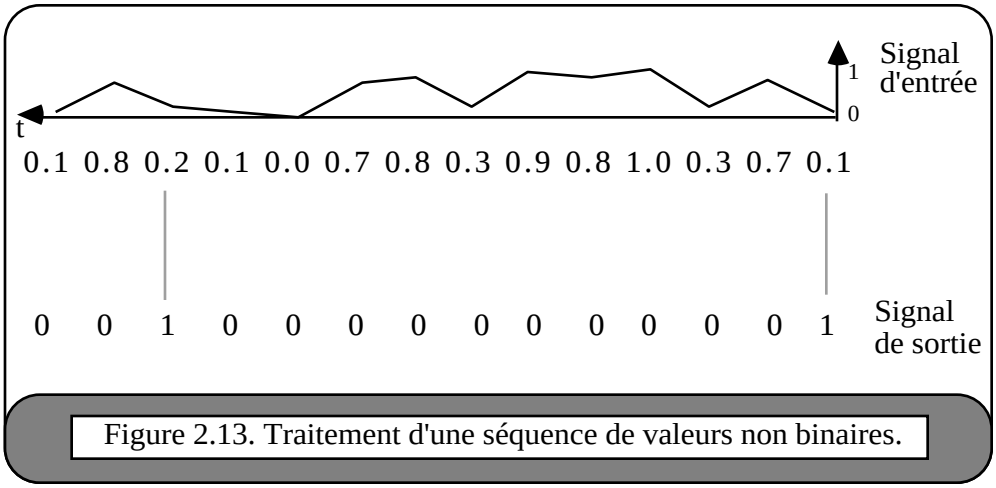

Figure 2.13. Traitement d'une séquence de valeurs non binaires.

La partie supérieure de la figure 2.13 montre la valeur du signal sur l'entrée du réseau en fonction du temps. Nous avons indiqué sous la courbe les valeurs exactes du signal. Il est à noter que la courbe montre une valeur continue dans le temps pour le signal d'entrée, alors que celui-ci est échantillonné. Chaque valeur d'échantillonnage est représentée par sa valeur exacte.

Sur la ligne inférieure est notée la valeur du signal de sortie rendue par le réseau, en correspondance avec l'entrée.

Conclusion

Cet exemple montre que la généralisation à des séquences de valeurs discrètes, autres que binaires, est réalisable. Par exemple, ayant appris à se comporter comme un automate détecteur de la séquence binaire 010, la machine séquentielle connexionniste est capable de reconnaitre une séquence telle que : 0.8 0.1 0.9 comme bonne. Ici, le modèle n'est plus à la recherche de la séquence binaire 010, mais d'une variation des entrées du type : valeur faible , valeur forte , valeur faible.

3.3 Apprentissage de type états contraints

Nous avons développé un second type d'apprentissage : l'apprentissage à états contraints. En effet, l'apprentissage de type états supervisés impose que le superviseur choisisse le codage des états internes de la machine séquentielle connexionniste. Ce codage est plus ou moins arbitraire et donc plus ou moins adapté au problème traité. Certains codages conviennent mieux que d'autres à l'apprentissage de diverses tâches séquentielles.

L'apprentissage à états contraints laisse quelques libertés au réseau neuronal dans le choix de la représentation des états internes. Le codage des états internes n'est pas complètement fixé par le superviseur. Le superviseur définit des intervalles de valeurs [Jordan 86] pour le codage des états internes. Ceci est mieux illustré sur un exemple. Dans le cas de la machine séquentielle connexionniste réalisant le ou-exclusif séquentiel, les états internes étant codés sur deux neurones, un codage des états internes contraints peut-être le suivant :

- E0 -> [-1/2, +1/2] - E1 -> [-1, 0] - E2 -> 1
[-1/2, +1/2] 1 [-1, 0]

Avec ce type de codage, lorsque l'état E0 est à apprendre, toute valeur des neurones de sortie de la fonction de transition appartenant à [-1/2, +1/2] est considérée comme juste. Il n'y a donc pas d'erreur et par conséquent pas de modifications des poids du réseau. Par contre, si la valeur d'un neurone d'état interne est hors de l'intervalle, une erreur est générée. L'erreur est la différence entre le milieu de l'intervalle désiré et la valeur calculée pour la cellule. Dans le cas ou l'intervalle est de la forme [-1, 1], il n'y a jamais d'erreur générée au cours de l'apprentissage.

Les résultats sur un problème de détection de séquences de longueur 3 mettent en avant la génération détats internes multiples qui permettent, par exemple, de tracer le passé. D'autre part, il est possible de réaliser une réduction de la table des états.

3.3.1 Algorithme d'apprentissage

L'apprentissage nécessite deux applications de la procédure de rétropropagation. Celle effectuée sur le réseau de transition est modifiée pour tenir compte des contraintes. L'erreur est calculée selon la procédure décrite au paragraphe précédent.

Les deux impératifs liés à l'emploi de l'apprentissage à états supervisés sont toujours présents, bien que le second soit atténué :

(1) Le superviseur doit connaitre le graphe d'états de l'automate à synthétiser .

(2) Le superviseur impose des contraintes sur le codage des états internes par la machine séquentielle connexionniste.

3.3.2 Exemple : le détecteur de la séquence de formes 010

Nous avons testé l'apprentissage à états contraints sur le même exemple que celui utilisé pour l'apprentissage à état supervisé. Cependant, notre attention s'est focalisée sur le comportement du réseau réalisant la fonction de transition d'état. C'est à ce niveau que les apports de l'apprentissage à états contraints sont les plus riches.

La base d'exemple : reprend celle construite pour la machine séquentielle connexionniste à état supervisé appliquée à la détection de la séquence 010.

Codage : le codage des formes présentées à l'entrée de la machine reste le même que pour l'apprentissage à états supervisés. Par contre, le codage des états internes est le suivant :

- E0 -> [-1, +1] [-1, +1] 1 - E1 -> [-1, +1] 1 [-1, +1] - E2 -> 1 [-1, +1] [-1, +1]

Il s'agit d'un codage du type 1 parmi n, chaque état interne active sélectivement une des cellules de sortie, l'état des autres cellules de sortie est indifférent.

Architecture : Dans le cas présent, nous nous intéressons principalement au réseau réalisant la fonction de transition d'état. Il se compose de 4 neurones sur la couche d'entrée, 4 neurones cachés et 3 neurones de sortie. Les 3 neurones de sortie sont rebouclés sur 3 neurones de la couche d'entrée.

Apprentissage : après 4000 itérations d'apprentissage, le réseau suit correctement le graphe des états de l'automate (figure 9) en fonction des entrées. Le réseau a donc correctement appris la tâche. Une étude plus fine révèle que le réseau s'est stabilisé dans plusieurs configurations différentes pour E0 (2 configurations E0', E0''), E1 (2 configurations E1', E1''). Cependant, il y a une seule configuration pour E2. Les configurations obtenues sont les suivantes :

E0' -> -0.011	E0'' -> 0.081	E1' -> -0.037	E1'' -> -0.041	E2 -> 0.88
-0.019	-0.085	1	0.9	-0.081
1	1	0.081	0.01	0.053

Chacune des configurations adoptées est correcte, l'écart maximum entre la valeur obtenue et celle désirée est de 0.12 (soit 4%). Les deux autres cellules permettent de repérer les transitions utilisées entre les états. Ainsi, dans le cas du passage de l'état E2 à E0, il y a 2 transitions possibles selon que la valeur de l'entrée est 0 ou 1. Ce qui correspond à 2 configurations différentes pour E0 : E0' et E0''. Il y a donc repérage de la valeur d'entrée. Les configurations permettent aussi de repérer entre les états précédents. Ainsi, les 2 configurations proposées pour le codage de E1 permettent d'identifier l'état précédent (E0 ou E1). L'état E1 est atteint dans les deux cas avec la même valeur d'entrée 0.

La figure 2.14 montre le graphe d'état de la machine séquentielle connexionniste après apprentissage. Nous avons indiqué en icone le graphe d'état de l'automate générateur des séquences d'apprentissage.

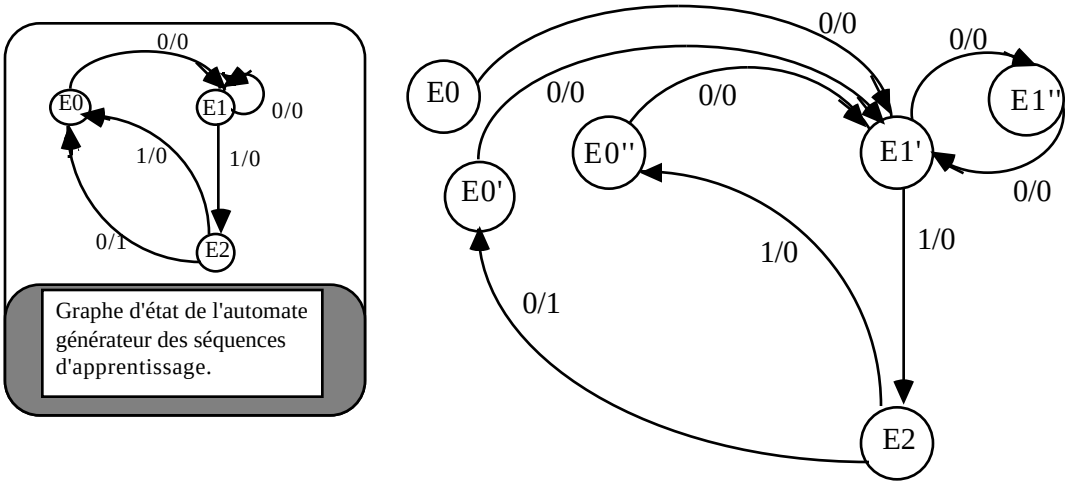

Figure 2.14. Graphe d'état de la machine séquentielle connexionniste, lors du problème de la détection de la séquence de formes 010 (apprentissage à états contraints).

Exemple de traitement d'une séquence après apprentissage :

	0	1	0	0	0	1	1	0	1	1	Séquence d'entrée
E0	E1	E2	E0	E1	E1	E2	E0	E1	E2	E0	Sortie désirée
	E1'			E1'			E1'				Codage adopté par le
					E1''						réseau, dans le cas de
			E0'								l'apprentissage à états
							E0''		E0''		contraints.
E0	E1'		E2	E0'	E1'	E1''	E2	E0''	E1'	E2	E0'' Sortie obtenue

Dans cette illustration, nous avons représenté sous chaque état interne désiré, le codage choisi par le réseau.

Conclusion : L'apprentissage à états contraints multiplie les configurations d'états internes. Il est parfaitement possible, en utilisant cette redondance, de retracer l'histoire du réseau. Nous pouvons recontruire la séquence des éléments d'entrée d'après la séquence de sortie obtenue. Ceci n'est pas possible dans le cas de l'étude des sorties de la fonction de transition d'états de l'automate détecteur équivalent.

3.3.3 Réduction du nombre d'états internes

L'un des problèmes liés à l'utilisation des réseaux neuronaux découle du choix hasardeux du codage des informations tant en entrée qu'en sortie. Ce codage peut parfois rendre l'apprentissage impossible, les formes étant incompatibles (dans notre cas, les états internes). Les possibilités de l'apprentissage à états contraints permettent d'envisager de réduire les cas de codage incompatibles. L'exemple suivant illustre cette plus grande indépendance de l'apprentissage vis à vis du codage.

Soit deux codages différents pour le même état E1. Au sein des séquences d'apprentissage, nous avons sélectionné, au hasard, les échanges entre les deux codages E1 et E3. Le graphe d'état de l'automate générateur des séquences d'apprentissage est représenté en icone de la figure 2.15. Ainsi, la séquence d'états internes suivante :

E0 E1 E2 E0 E1 E1 E2 E0 E1 E2 E0 E0 E1 E1 E1

devient : E0 E1 E2 E0 E3 E1 E2 E0 E1 E2 E0 E0 E3 E3 E1

C'est cette séquence que nous allons soumettre au réseau. Les permutations réalisées entre E1 et E3 ont été soulignées.

Choisissons le codage suivant pour les 4 états internes E0, E1, E2, E3 :

-E0 ->	[-1, +1]	- E1 ->	[-1, +1]	- E2 ->	[-1, +1]	- E3 ->	1
	[-1, +1]		[-1, +1]		1		[-1, +1]
	[-1, +1]		1		[-1, +1]		[-1, +1]
	1		[-1, +1]		[-1, +1]		[-1, +1]

Apprentissage : après 4000 itérations d'apprentissage, le réseau suit correctement le graphe des états de l'automate décrit figure 2.15. La question que nous posons est : la configuration adoptée par le réseau pour représenter E1 est-elle semblable à celle utilisée pour représenter E3 ? Est-il possible au sein du graphe des états de réunir E1 et E3 ?

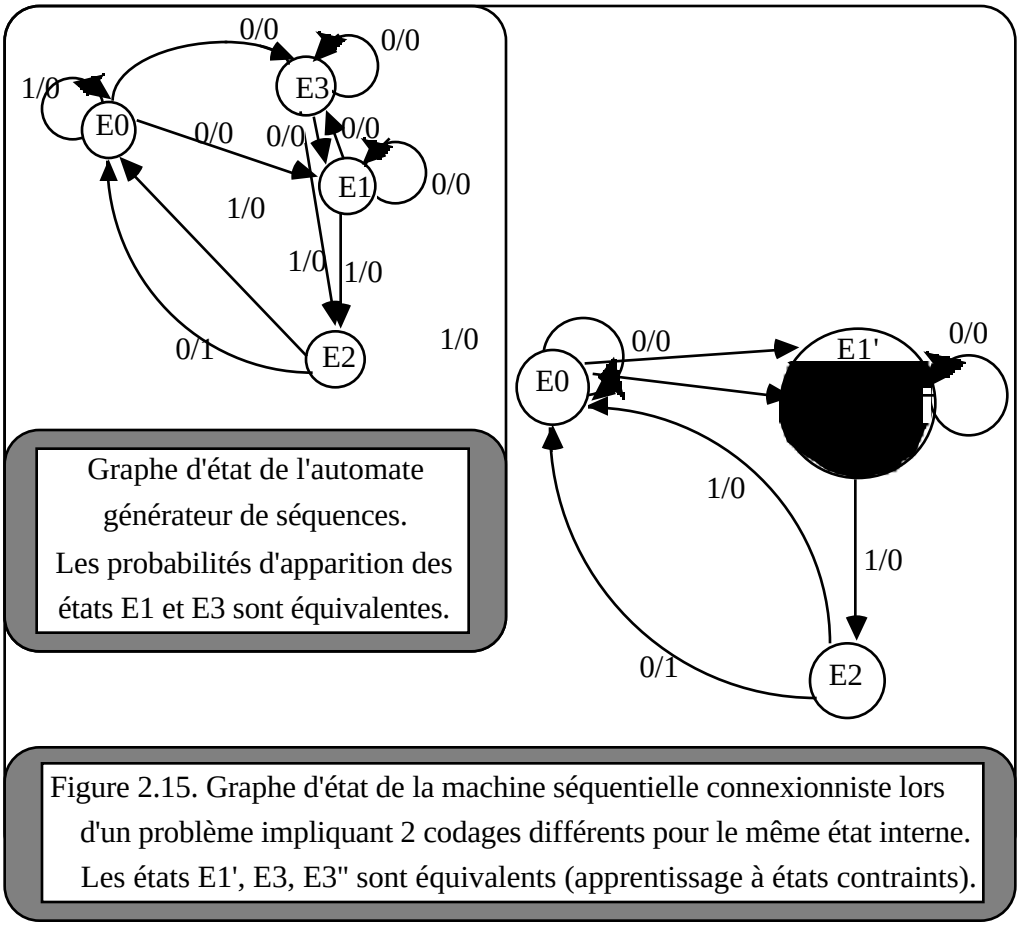

Figure 2.15. Graphe d'état de la machine séquentielle connexionniste lors d'un problème impliquant 2 codages différents pour le même état interne. Les états E1', E3, E3" sont équivalents (apprentissage à états contraints).

Après expérimentation, nous obtenons :

E0 E1 E2 E0 E3 E1 E2 E0 E1 E2 E0 E0 E3E3 E1 Sortie désirée
E1' E1' Codage adopté par le
E1'' E1'' réseau, dans le cas de
E3' l'apprentissage à états
E3'' contraints.
E1"

Dans cette illustration, nous avons représenté sous chaque état interne désiré, le codage choisi par le réseau. Les configurations obtenues sont les suivantes :

-E1'-> 0.32 - E1" -> 0.41 - E3' -> 0.4 - E3" -> 0.41
0.098 0.078 0.12 0.082
0.42 0.51 0.5 0.51
0.32 -0.019 0.003 -0.016

Chacune des configurations adoptées est correcte. Les cellules les plus actives correspondent au codage respectif de E1 et E3. Nous notons que E1'' = E3' = E3''. Le réseau est capable de réduire le nombre des états internes si ceux-ci sont dupliqués. A cet effet, le réseau utilise la souplesse du codage des états contraints pour rassembler au sein d'un même codage les états dupliqués, tant que les probabilités d'apparition de chaque état sont équivalentes.

Conclusion : l'apprentissage à états contraints apporte une certaine souplesse au niveau du codage permettant :

- de tracer le passé en multipliant les configurations d'états internes,
- de réduire les possibilités de codages incompatiblese entre états internes générés et donc de multiplier les probabilités de convergence de l'algorithme.
- de rassembler dans une même représentation les états dupliqués.

Remarque : dans l'exemple présenté, le codage des états internes en 1 parmi n permet d'étudier facilement les propriétés de l'apprentissage à états contraints. Cependant, ces propriétés ne se limitent pas à un codage particulier.

3.4 Apprentissage à états non supervisés

Malgré l'apport de souplesse de l'apprentissage à états contraints au niveau du codage des états internes, il revient toujours à l'utilisateur de définir le graphe des états du problème à résoudre. Désirant dépasser cettre limitation, nous proposons un algorithme d'apprentissage baptisé à états non supervisés. L'utilisateur n'a pas spécifié le graphe des états. L'apprentissage est effectué sur la machine séquentielle connexionniste globale (et non découpée en deux sous réseaux) à partir des exemples d'entrée/sorties.

L'apprentissage ne modifie pas automatiquement les deux fonctions de transition et de sortie. On privilégie une modification de la fonction de sortie. Cependant, dans le cas où le problème soumis est effectivement de nature séquentielle, alors la fonction de transition est modifiée. Cet algorithme essaye, autant que possible, de ne modifier qu'une des deux fonctions pour chaque forme de la séquence présentée.

3.4.1 Algorithme d'apprentissage

- Initialisation : état stable sur le réseau de transition $s(t-1) = s(t)$ et $i(t) = 0$.

Pour la forme i(t) de la séquence :

- Propagation : comportement décrit par les équations :

  $\delta(i(t), s(t-1)) = s(t)$ fonction de transition,

  $\lambda(i(t), s(t-1)) = o(t)$ fonction de sortie.

- Apprentissage : soit $od(t)$ la sortie désirée. Il y a 2 cas possibles :

  1) $od(t) = o(t)$ Passage à la forme suivante de la séquence.

  2) $od(t) \neq o(t)$ Alors 2 cas possibles:

Modification de la fonction de sortie

On cherche à obtenir $od(t) = o(t)$ par modification de la fonction de sortie. Ceci est réalisé par application de la rétropropagation de gradient pendant un nombre $\leq N$ d'itérations d'apprentissage (N nombre fini). De plus, comme il ne faut pas perdre les données déjà acquises, les itérations d'apprentissage sont aussi effectuées sur l'ensemble des exemples traités.

Modification de la fonction de transition

Après N d'itérations d'apprentissage : $od(t) \neq o(t)$. Une modification de la fonction de sortie n'a pas été suffisante. Il faut modifier la fonction de transition. Une configuration $s'(t)$, telle que $l(i(t), s'(t)) = od(t)$ est choisie. Connaissant la valeur de s', nous appliquons sur la fonction de transition l'algorithme de la rétropropagation de gradient pour obtenir $d(i(t), s(t-1)) = s'(t)$.

La détermination de l'état interne désiré $s'(t)$ est réalisée par un algorithme qui rappelle celui de la rétro-propagation des états désirés [Le Cun 87c] :

Soient $s(t) = (..., xj, ...)$; $s'(t) = (..., x'j, ...)$; $o(t) = (..., xi, ...)$

La règle de détermination des états désirés $x'j$ est la suivante (il y a rétropropagation des valeurs désirées depuis la sortie vers l'entrée du réseau) :

$$x'j = xj + k.(\sum_i ei.wij)$$

Où j est un neurone prédecesseur du neurone i, k est une constante, wij est le poids de la connexion entre le neurone j et le neurone i,

pour les neurones de sortie : $ei = (xdi - xi)$, xdi est la valeur d'activation désirée,

pour les autres neurones : $ei = (x'i - xi)$.

Il faut éventuellement itérer plusieurs fois l'algorithme de détermination de $s'(t)$ et relancer une phase d'apprentissage sur la fonction de sortie si la détermination de $s'(t)$ conduit à des valeurs impossibles.

N est un nombre d'itérations qui devrait être infini. En effet, le nombre d'itérations nécessaires à la convergence de l'algorithme de la rétropropagation n'est pas connu. On le fixe arbitrairement. De ce fait, on court le risque de croire que le problème est de nature séquentiel, alors que c'est faux.

Base d'exemples d'apprentissage

Les exemples utilisés pour l'apprentissage supervisés sont de la forme suivante :

$\{[(\text{Entrée}) \rightarrow (\text{Sortie désirée})]_j\}_i$ pour la forme j de la séquence i.

Ci-dessous sont décrits les exemples d'apprentissage dans le cas du ou-exclusif séquentiel (4 séquences de longueur 2) :

$\{[(0) \rightarrow (0)]_1,$

$[(1) \rightarrow (1)]_2\}_1$

$\{[(1) \rightarrow (0)]_1,$
$[(0) \rightarrow (1)]_2\}_2$
$\{[(0) \rightarrow (0)]_1,$
$[(0) \rightarrow (0)]_2\}_3$
$\{[(1) \rightarrow (0)]_1,$
$[(1) \rightarrow (0)]_2\}_4$

3.4.2 Exemple : le ou-exclusif séquentiel

Conclusion

Le terme d'apprentissage non supervisé est employé pour illustrer le fait que l'utilisateur n'a pas à définir le codage des états internes. Il doit cependant choisir le nombre de neurones utiliser pour représenter les états internes (nombre de neurones de la couche de sortie du réseau de transition). L'apprentissage est réalisé par l'application d'une procédure de rétropropagation, soit sur le réseau de sortie, soit sur le réseau de transition. La priorité est donné à une modification du réseau de sortie. Dans le cas d'une modification du réseau de transition, préalablement à la rétropropagation, un algorithme de détermination de l'état interne désiré est appliqué. Les résultats sur un petit problème de reconnaissance de séquences de longueur 3 montre qu'il est possible d'acquérir le comportement d'un automate sans connaissance a priori du graphe des états. Il est aussi possible de déterminer a priori de la nature séquentielle ou combinatoire du problème soumis.

3.5 Conclusion

Nous avons présenté et illustré les propriétés des différents algorithmes d'apprentissage sur le modèle de la machine séquentielle connexionniste sur des petits problèmes pédagogiques. Ces propriétés de généralisation et de flou sur la définition des états internes interdisent, comme l'a mentionné Kohonen [Kohonen 87b], la réalisation d'une machine séquentielle connexionniste strictement équivalente à une machine séquentielle. Ce n'est pas notre but. En effet, les études menées ont montré que ce sont ces propriétés mêmes qui conduisent à des domaines d'application différents et nouveaux. Citons pour mémoire la possibilité de généraliser l'apprentissage à des séquences non apprises (à valeurs discrètes, alors que l'apprentissage est réalisé avec des valeurs binaires), de tracer le passé par multiplication des états internes, de réduire la table des états, de réaliser l'apprentissage sans connaissance à priori du graphe des états, de déterminer à priori la nature séquentielle d'un problème.

4 Conclusion

___La machine séquentielle connexionniste fait référence explicitement à la machine séquentielle par l'utilisation d'une fonction de transition et d'une fonction de sortie. Le comportement est celui d'une machine déterministe synchrone. Les différents apprentissages présentés pour ce modèle diffèrent par la connaissance qu'il faut posséder à priori du problème, en fait du graphe des états. L'ordre de présentation des algorithmes d'apprentissage découle d'une connaissance de moins en moins importante du problème en contrepartie d'une complexification des procédures d'apprentissage. Le domaine d'application de chaque type d'apprentissage est spécifique :

- Apprentissage à états supervisé : il faut connaitre le graphe des états. Il y a possibilité de généraliser l'apprentissage à des séquences de valeurs inconnues.

- Apprentissage à états contraints : il faut connaitre le graphe des états. Il y a possibilité de réduction de la table des états. La multiplicité des états internes permet de tracer le passé.

- Apprentissage à états non supervisé : la connaissance du graphe des états n'est pas nécessaire. Il y a possibilité de déterminé à priori la nature séquentielle de l'application traitée.

CHAPITRE III

LE LOGICIEL SACREN *

1 Présentation générale

SACREN : Système d'Aide au Choix d'un Réseau de Neurones est né du recours obligatoire à l'expérimentation dans le domaine des réseaux de neurones artificiels pour le choix du modèle connexionniste le plus adapté à une application particulière. Sous la dénomination générale de système connexionniste, modèle neuromimétique ou plus communément réseau de neurones artificiels se regroupe un grand nombre de modèles de réseaux différents. Les spécifications applicatives de chaque architecture de réseau ne reposent pas sur des résultats théoriques (sauf pour les plus simples : Perceptron [Minsky 88]). Le recours à l'expérimentation s'avère être le moyen le plus approprié pour choisir le type de réseau à adopter pour une application déterminée.

---

* Projet réalisé dans le cadre d'un contrat entre le LERI et l'ANVAR [Touzet 89b]

SACREN fourni un environnement permettant à l'utilisateur/développeur de tester facilement les diverses possibilités neuromimétiques pour une application donnée. SACREN offre ainsi la possibilité de sélectionner une structure de réseau de neurones plus adaptée au problème industriel. Dans ce but, SACREN permet de simuler tous les types de réseaux neuronaux, certains d'entre eux ont fait l'objet d'une implantation informatique : les réseaux multi-couches et la rétro-propagation de gradient, les mémoires associatives linéaires et les cartes auto-organisatrices de Kohonen.

SACREN offre des possibilités que l'on ne retrouve pas au niveau des simulateurs des réseaux de neurones commercialisés :

Une grande facilité de conceptualisation :

- Une description informatique existe pour chaque neurone et chaque synapse (structure de données).
- Simulation événementielle (comportement dynamique).

Des facilités pour la modification :

- Toutes les topologies de réseaux sont autorisées.
- Chacun des neurones et des synapses peut être doté d'un comportement particulier.
- Il a été développé sur station de travail graphique Apollo DN 3000, ce qui permet de disposer d'un environnement de programmation complet tant au niveau matériel que logiciel. On peut citer par exemple :
    - Processeur rapide 68020,
    - Ecran graphique haute résolution (19"),
    - Espace disque 170 Méga-octets,
    - Souris disponible,
    - Compilateur, éditeur de liens et debuggeur de qualité professionnelle,
    - Système d'exploitation performant : Unix Système V,
    - Ergonomie d'utilisation : multi-fenêtrage (Dialogue), éditeur pleine page.

2 Session sur SACREN

Une session typique est décrite figure 3.1. Tout d'abord, il faut choisir la structure du réseau, sa taille et l'algorithme d'apprentissage qui va être utilisé. Puis, la phase d'apprentissage sur la base d'exemples est exécutée. L'étape suivante est l'utilisation du réseau qui permet de valider ou d'invalider les choix réalisés lors des phases précédentes.

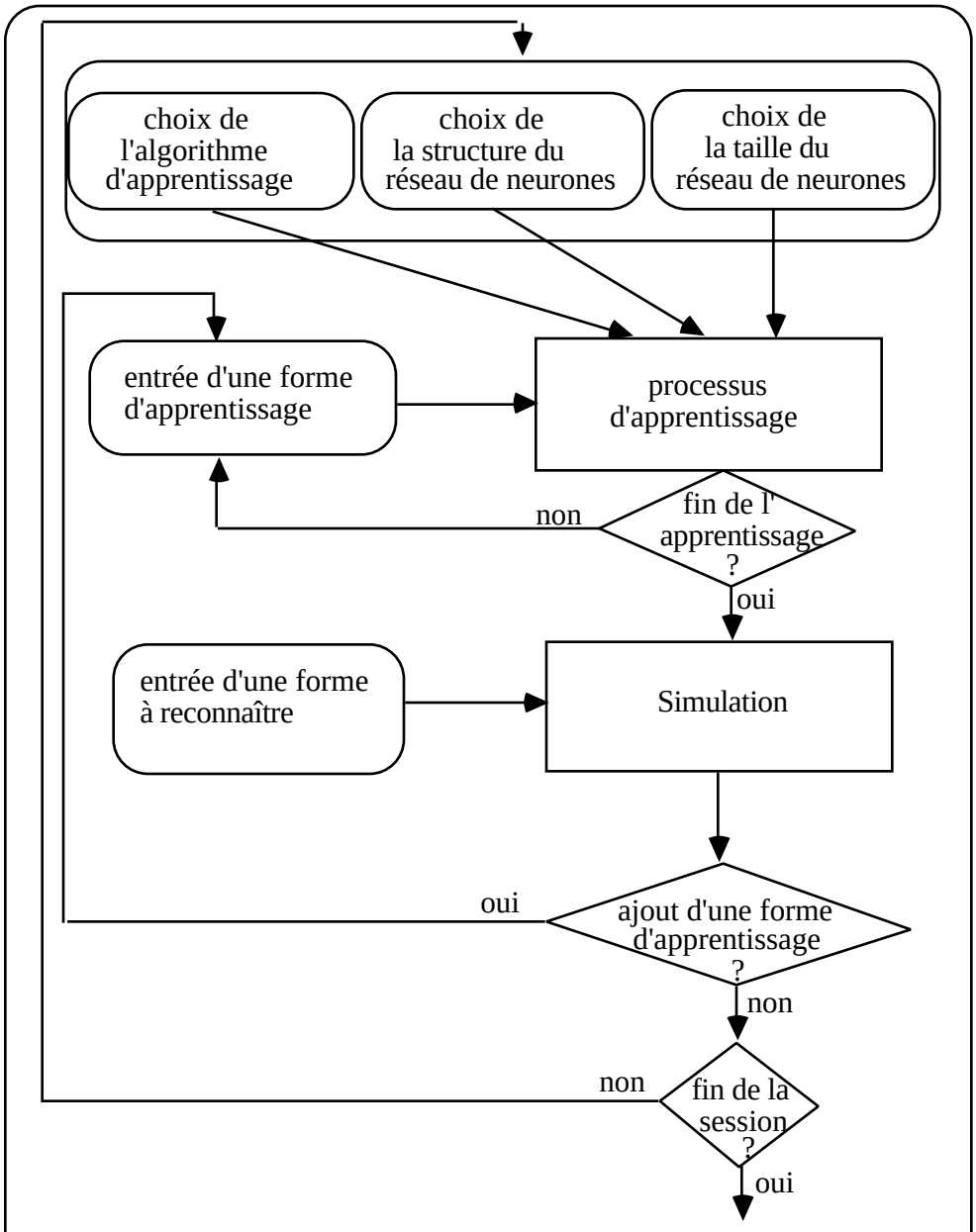

Figure 3.1. Organigramme d'une session d'utilisation du système SACREN

3 Architecture générale du système

Le principal critère guidant le développement de SACREN a été la flexibilité. Le simulateur est basé sur les concepts de cellules et de synapses (connexions). Il permet ainsi la construction et la simulation d'une grande variété de réseaux (chaque cellule possède son comportement propre, tout type de schéma de connexions peut être spécifié).

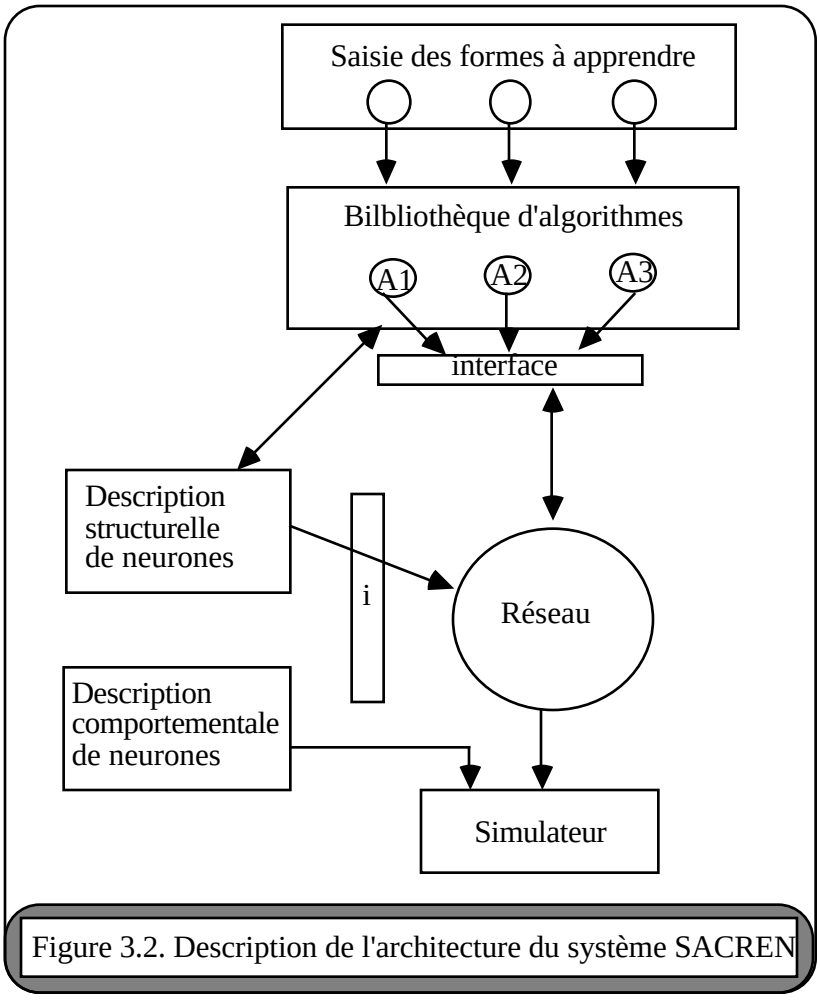

Figure 3.2. Description de l'architecture du système SACREN

La figure 3.2 décrit l'achitecture du système. Le module désigné sous le terme de description structurelle de neurones regroupe les utilitaires permettant de construire l'architecture du réseau de neurones : les données (schéma des connexions) utilisées pour la simulation. Le second module de description comportementale de neurones a pour objet la spécification du comportement de chacune des cellules. La bibliothèque des algorithmes regroupe différentes procédures d'apprentissage pour le réseau de neurones : telle que la rétro-propagation de gradient, … La procédure d'apprentissage travaille à partir d'une base d'exemples fournis par le module de saisie des formes à apprendre. Le module simulateur est composé d'un simulateur événementiel à événements discrets qui génère, classe et traite les

événements consécutifs à l'application sur le réseau d'une forme en entrée. Les interfaces sont des procédures proposant à l'utilisateur, sous forme de menu, toutes les actions impliquées dans le déroulement d'une session. Chacun de ces modules est repris plus en détail dans les paragraphes suivants.

Description structurelle de neurones

Il s'agit d'un module évolutif destiné à décrire l'architecture de chacun des modèles implantés. On entend par modèle implanté, tout modèle dont l'algorithme d'apprentissage a été décrit et ajouté à la bibliothèque des algorithmes. Ce module permet à la date actuelle la construction de réseaux à une ou plusieurs couches, sans connexion intracouche et à connexion complète intercouches, avec des options pour décrire des réseaux bouclés (boucles depuis la couche de sortie ou la couche cachée), des cartes topologiques de Kohonen et des mémoires auto ou hétéro-associative. De plus, le stockage de l'architecture du réseau dans des fichiers aux normes ASCII permet à l'utilisateur une modification aisée des caractéristiques, l'ajout ou l'annulation de certaines synapses, la spécification de valeurs particulières à certains poids, ... En fait tous les composants relatifs à la description structurelle d'un réseau sont accessibles et modifiables par l'utilisateur.

Description comportementale de neurones

Il existe plusieurs fonctions différentes pour le calcul de la valeur de l'état du neurone:

- fonction signe,
- fonction sigmoïde avec prise en compte du passé : l'état précédent du neurone intervient dans le calcul,
- fonction sigmoïde classique.

Il est possible de rajouter de nouvelles fonctions. Chaque neurone peut avoir une fonction de transfert différente de celle des autres neurones.

Nous proposons aussi plusieurs variantes au niveau des informations transmises depuis le neurone vers les autres neurones. Ceci décrit le comportement du réseau. Toutes les cellules du réseau transmettent la même information :

- propagation de l'état du neurone,
- propagation de la variation d'état (état précédent - état présent).

Bibliothèque des algorithmes

Parmi les algorithmes d'apprentissage implanté, nous recensons les cartes auto-organisatrices, les mémoires associatives et plusieurs versions de la rétropropagation de gradient. Chacune relève de la même philosophie générale, mais se distingue soit par le

comportement des cellules (fonction de transfert), soit par l'architecture du réseau (réseaux bouclés).

Saisie des formes à apprendre

Le passage de la forme brute (pixels de l'image par exemple) à sa représentation sur la couche d'entrée du réseau dépend principalement des caractéristiques architecturales du réseau. L'apprentissage supervisé impose que l'on dispose pour chaque forme d'entrée de la forme de sortie désirée. Il est nécessaire d'avoir une procédure de codage qui permette le passage de la forme desirée au codage sur la couche de sortie du réseau. D'autre part, la base de données est fonction de l'application envisagée. Ainsi, une application de reconnaissance des formes ne requiert comme exemples d'apprentissage et de test que des images (ensemble de pixels), tandis qu'une application de diagnostic impose que cette même base de données soit composée d'un grand nombre de couples (symptômes, diagnostic associé). La structure de données informatique pour mémoriser les bases d'exemples de ces deux types d'application est incomptatible, bien que le réseau de neurones reste identique à tous points de vue. Le type d'application envisagée engage une certaine spécificité des outils de saisie de la base d'exemples.

Interfaces

L'interface permet, en sélectionnant au sein d'un menu général d'une vingtaine d'options, de réaliser toutes les fonctions de base du système : apprentissage, simulation, affichage de la table des cellules, affichage de la table des synapses, quitter le système, sauvegarde de l'état du réseau, affichage des formes d'entrée/sortie, acquisiton des formes d'entrée et de sortie, saisie interactive de forme d'entrée, construction d'un architecture de réseau, acquisition d'un réseau existant, changement de valeurs des paramètres de la rétropropagation.

Les fonctions liées à la vérification du bon fonctionnement du simulateur (debug) sont incluses dans chacune des procédures. Il existe aussi des procédures d'affichage des valeurs des tables des neurones et des synapses.

Les fonctions liées au test des performances des réseaux simulés (test) répondent à l'objectif de notre logiciel : "destiner aux laboratoires et à l'industrie, il doit principalement permettre de tester différents algorithmes d'apprentissage afin de déterminer expérimentalement celui qui est le mieux adapté à leur problème". Les outils de test sont regroupés dans une procédure qui calcule l'écart entre la réponse fournie par le réseau et celle desirée. Cet écart est en fait une mesure des performances du système. Il est possible de mesurer les performances du système à n'importe quel instant de la phase d'utilisation ou de la phase d'apprentissage. [Chappaz 88].

Structure de données

La principale structure de données est un tableau des cellules (neurones) comprenant une listes des connexions (synapses) au départ de celle-ci. Un second tableau regroupe les synapses avec le délai qui leur est attaché ainsi que le numéro du neurone destination. Les neurones et les synapses contiennent de plus diverses informations comme, pour les neurones, le potentiel, le type de la fonction de transfert, la somme pondérée des entrées et quelques valeurs relatives au passé de la cellule.

3.1 Le simulateur

SACREN fait appel à la simulation dirigée par les événements. La simulation événementielle présente de multiples avantages par rapport au calcul matriciel (matrice des poids) classiquement utilisée [McClelland 88], [HNC 89]. Rappelons brièvement ici, les trois principaux avantages liés à son emploi :

- Gain en facilité de conceptualisation : les neurones et synapses sont effectivementreprésentés comme des entités au sein du modèle simulé.

- Gain de temps de calcul : seules les cellules ayant changé d'états vont créer un événement, ainsi le minimum de calcul est effectué. Rappelons que le calcul matriciel oblige de recalculer à chaque interaction la valeur de tous les neurones du réseau. Ceci est particulièrement pénalisant dans le cadre des réseaux multicouches sans boucle où les couches sont activées les unes après les autres (et une seule à la fois).

- Gain de place mémoire : la structure de données décrivant le réseau est minimale, en effet seules les cellules et les connexions existantes sont décrites. La technique classique (matricielle) impose de manipuler des matrices de connexions très creuses. Ainsi dans le cas d'un réseau de neurones multicouches à connexions complète intercouches sans boucles comprenant n cellules, la taille de la matrice est n^2 alors qu'il existe environ $n^2/10$ connexions effectives. Pour des réseaux degrandes dimensions, le gain de taille mémoire est considérable.

Evènements : la simulation du fonctionnement des réseaux de neurones nécessite trois types différents d'événements que nous avons baptisés : événement_synapse, événement_neurone, événement_zéro. Nous décrivons les actions réalisées par chacun de ces événements.

- L'événement_synapse est chargé de transmettre depuis un neurone source jusqu'à un neurone destination, la valeur de l'état du neurone source pondérée par le poids de la connexion. De plus, cet événement permet l'introduction explicite au niveau de la synapse du concept de temps. Il est possible d'attribuer un retard à la transmission entre deux neurones. Cet apport de la simulation événementielle n'est pas encore exploité au sein des modèles neuromimétiques développés actuellement. Chaque événement_synapse crée un événement_neurone.

- L'événement_neurone assure la mise à jour de la somme pondérée du neurone.

- L'événement_zéro assure le synchronisme de la mise à jour du traitement des neurones.

Echéancier : c'est la structure dans laquelle sont stockés les événements. Du fait de la manipulation explicite du temps dans notre modèle de réseau de neurones, à chaque événement est associé une date. L'échéancier doit assurer le classement des événements en fonction de leur date afin que soit traité en premier l'événement le plus proche dans le temps. La structure informatique de l'échéancier est présentée figure 3.3.

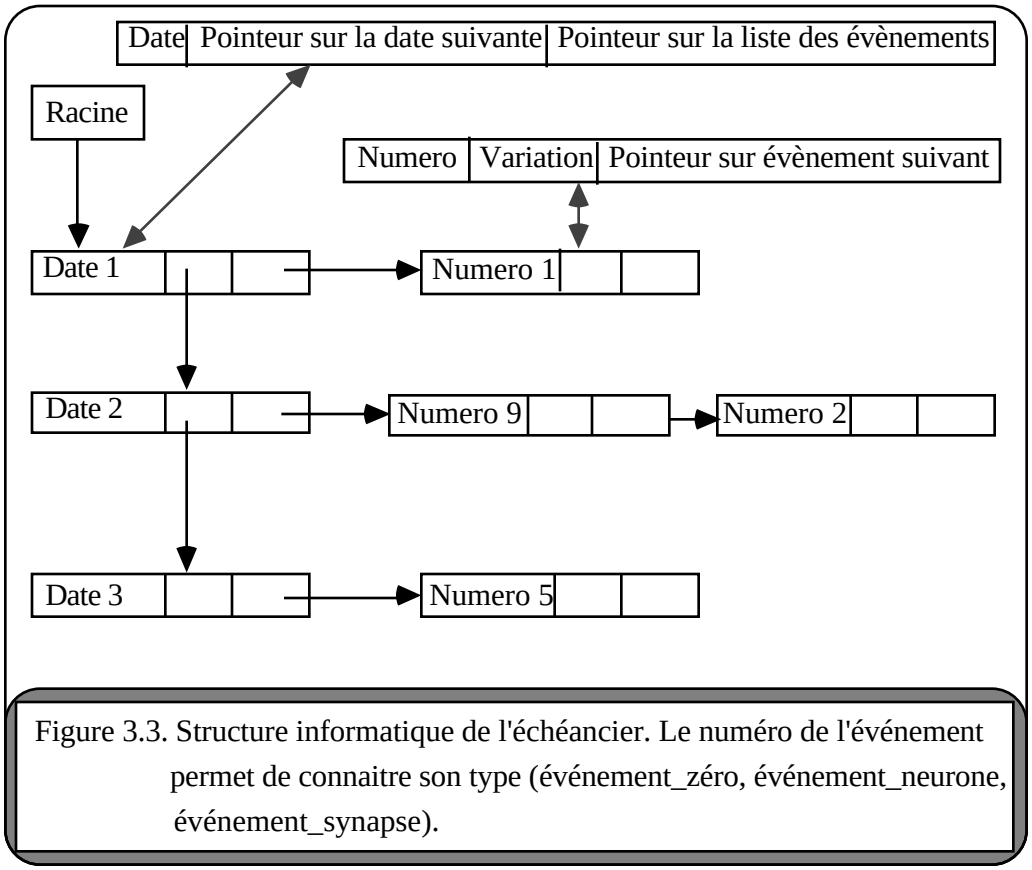

Figure 3.3. Structure informatique de l'échéancier. Le numéro de l'événement permet de connaitre son type (événement_zéro, événement_neurone, événement_synapse).

4 Quelques idées pour un futur simulateur

L'idée initiale qui a guidé le développement de SACREN est la simulation du plus grand nombre possible de modèles connexionnistes au sein d'un même outil. Ceci dans l'optique d'une éventuelle association entre réseaux, d'une utilisation du même formalisme et d'un même environnement de description (voir des mêmes variables). Cet objectif n'est que partiellement atteint, car les modèles de réseaux sont effectivement trés différents les uns des autres. Réaliser la phase de propagation de divers modèles neuronaux avec un simulateur unique est possible. C'est la partie que l'on peut qualifier de plus simple. Mais certaines fonctions sont beaucoup plus difficiles à rassembler au sein d'un même simulateur. Un grand nombre de fonctions dépendent fortement du modèle, par exemple : l'apprentissage, l'acquisition d'image, le test, la vérification, l'initialisation.

A l'usage, nous avons pu vérifier que la gestion du logiciel devient de plus en plus complexe en fonction du nombre de modèles de réseaux implantés, jusqu'à rendre impossible toute mise à jour, modification ou vérification. Toute manipulation d'un modèle se répercute sur l'ensemble du système.

Nous concluons en notant qu'il est chimérique de vouloir au sein d'un même programme simuler tous les modèles connexionnistes. Ces modèles sont à la fois trés nombreux et trés variés. Citons pour exemple le jeu de la vie et les réseaux sémantiques, deux modèles à la frontière du connexionnisme. Il faut cependant, dans l'intérêt de l'utilisateur conserver l'idée d'une interface commune, d'une même action des commandes, etc... Ceci n'est pas, loin s'en faut, un souci des logiciels commercialisés à ce jour dans ce domaine.

D'autre part, l'utilisation de la simulation événementielle montre une réduction du nombre de calculs, un gain en terme d'espace mémoire et la possibilité de manipuler des retards au niveau des connexions, bien qu'encore inemployée. Ces avantages ne sont pas remis en cause.

CONCLUSION GENERALE

Le modèle de la machine séquentielle connexionniste présenté dans ce mémoire constitue une évolution logique des approches connexionnistes du séquentiel. Inspiré par le concept de machine séquentielle tel que le défini la théorie des automates, il représente une généralisation d'une classe de modèles : les réseaux séquentiels. La machine séquentielle connexionniste ne permet pas seulement de décrire les différentes approches dans un cadre plus général, mais aussi de prédire et d'expliquer les applications possibles de chacune [Park 88].

L'étude de ce modèle sur petits problèmes pédagogiques montre des propriétés nouvelles par rapport à la machine séquentielle :

- Généralisation de la reconnaissance à des séquences de valeurs discrètes alors que l'apprentissage a été réalisé sur des valeurs binaires.
- Réduction automatique de la table des états.
- Possibilité de retracer l'histoire des séquences soumises.
- Synthèse de l'automate sans connaissance à priori du graphe des états.
- Détermination automatique de la nature séquentielle ou combinatoire d'un problème.

Des applications plus importantes de ce modèle sont envisageables, en paticulier dans les domaines suivant :

- Prédiction d'éléments de séquences.
- Classification de séquences.
- Génération de séquences.

Prédiction : La tâche est de prédire les éléments successifs de la séquence. Par exemple, dans le cas de l'apprentissage d'une grammaire artificielle [Servan-Schreiber 88], le réseau doit suivre l'automate de la grammaire. A chaque lettre présentée, le réseau passe dans l'état suivant. On s'intéresse particulièrement au cas où deux instances de la même lettre conduisent à des états différents et de fait à des prédictions différentes pour les successeurs.

Classification/reconnaissance : Il s'agit pour le réseau de classer les séquences présentées. Ainsi, le réseau de neurones développé par [Liu 89], aprés l'apprentissage d'exemples issues d'une grammaire, classe les séquences comme appartenant ou non à la grammaire. Il est possible d'envisager de classer les séquences en de plus nombreuses classes. C'est le cas dans la reconnaissance de la parole, où le réseau a pour tâche d'associer à une séquence de phonèmes le mot correspondant [Jordan 86].

Génération : Il s'agit de produire de nouvelles séquences en se basant sur celles apprises. Par exemple, dans le cas de la composition musicale [Todd 88], le réseau de neurones compose de nouvelles séquences mélodiques à partir des mélodies apprises. Dans le cas d'un réseau conforme au modèle de Jordan, à chaque séquence correspond un vecteur de plan, il est possible de produire de nouvelles séquences en spécifiant un nouveau vecteur de plan, par exemple une combinaison des vecteurs de plans utilisés durant l'apprentissage.

Dans tous les domaines d'applications précédement cités, le fonctionnement du réseau de neurones reste le même. Par contre, l'interprétation des résultats varie. Dans le cas de la classification, la sortie calculée par le réseau est comparée avec la sortie désirée, à la présentation de chaque élément de la séquence. Dans le cas de la prédiction, la séquence est considérée comme un tout, la sortie calculée par le réseau est comparée avec la sortie désirée à la fin de la présentation de la séquence. Dans le cas de la génération, la sortie obtenue est considérée comme telle. Elle n'est pas comparée avec une valeur désirée.

Sous un aspect recherches, les travaux que nous avons réalisés sur les machines séquentielles connexionnistes constituent une première approche de l'étude d'architectures hiérarchisées de réseaux de neurones avec des techniques d'apprentissage ad-hoc en vue de résoudre des problèmes spécifiques. La résolution d'un problème par un réseau de neurones consiste à trouver par apprentissage la bonne relation d'entrée/sortie. Lorsque cette relation est complexe, le problème doit être décomposé en sous-problèmes terminaux. Chacun des comportements des sous-problèmes terminaux est réalisé par un réseau de neurones, que nous

appelons réseau de base. Il faut donc déterminer, pour un problème complexe donné, d'une part la structure du système et d'autre part, les comportements que doivent réaliser chacun des réseaux de bases.

Dans l'approche que nous avons présentée, la structure est fixée volontairement comme celle d'une machine séquentielle et nous avons étudié les possibilités d'apprentissage hors contexte si l'on connait pour chacun des réseaux de base (réseaux de transition et de sortie) le comportement à réaliser, ou en contexte à partir de la seule relation globale d'entrée/sortie. Les apprentissages à états supervisés et à états contraints sont hors contexte, l'apprentissage non supervisé est en contexte.

Une suite logique de ces travaux sur les machines séquentielles connexionnistes serait d'étudier différentes structures élémentaires (série, parallèle) comportant un faible nombre de réseaux. Les réseaux de base sont des réseaux multicouches, des cartes auto-organisatrices, des machines de Boltzmann, ... Pour ces différentes situations, il faut étudier les techniques d'apprentissage adaptées aux interconnexions considérées. Une deuxième étape est d'étudier les structures arborescentes des différents types de réseaux de base précédemment cités et de développer des techniques d'apprentissage adaptées à ces structures particulières.

REFERENCES BIBLIOGRAPHIQUES

[Allen 88] R. B. Allen, Connectionist State Machines, Report ARA 88-300, Bellcore, Morristown, NJ, USA, November 1988.

[Almeida 88] L.B. Almeida, "Backpropagation in perceptrons with feedback," in NATO ASI Series Vol F 41, Neural Computer, Paris 1988.

[Chappaz 88] E. Chappaz et N. Giambiasi, "STALINN : Statistical module for Testing Learning phase In Neural Nets," Neuro-Nîmes 88, Nîmes, France, novembre 1988.

[Decamp 88] E. Decamp et B. Amy, "Neurocalcul et réseaux d'automates, le point sur les recherches et les applications," EC2, 1989.

[Dehaene] S. Dehaene, J.P Changeux. and J.P. Nadal, "Neural networks that learn temporal sequences by selection," Proceedings of the National Academy of Sciences USA, Vol. 84 : 2727-2731, May 1987, Biophysics.

[Elman 88] J. L. Elman, "Finding Structure in Time," Center for Research in Language Technical Report 8801, University of California, San Diego, April 1988.

[Gallant 88] S. I. Gallant and D. J. King, "Experiments With Sequential Associative Memories," Conf. Cognitive Science Society, Montreal, Canada, August 17-19, 1988.

[Gori 89] M. Gori, "An Extension of BPS," Neuro-Nîmes 89, Nîmes, France, novembre 1989, pp 83-93.

[Hartmanis 66] J. Hartmanis and R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice-Hall International series in Applied Mathematics, 1966.

[Hebb 49] D. O. Hebb, "The Organization of Behavior," New-York : Wiley, 1949.

[HNC 89] Hecht Nielsen NeuroComputers, "ANZA Plus : User's Guide and Neurosoft Documents," Release 2.2, May 1989.

[Jordan 86] M. I. Jordan, "Serial Order : Parallel Distributed Processing Approach," ICS Report 8604, Univ. of California, San Diego 92093, May 1986.

[Jordan 88] M. I. Jordan, "Supervised learning and systems with excess degrees of freedom" COINS Technical Report 88-27, MIT at Amherst, May 1988.

[Karna 89] K. N. Karna and D. M. Breen, "An Artificial neural networks tutorial : part 1 - basics," The International Journal of Neural Networks Research and Applications, Learned Information, Oxford, Vol. 1, No. 1, January 1989, pages 4-23.

[Kleinfeld 86] D. Kleinfeld, "Sequential state generation by model neural networks," Proceedings of the National Academy of Sciences USA, Vol. 83 : 9469-9473, December 1986, Biophysics.

[Kohavi 78] Z. Kohavi, Switching and Finite Automata Theory, Second Edition, Computer Science Series, McGraw-Hill, 1978.

[Kohonen 87a] T. Kohonen, Self-Organization and Associative Memory, second edition, Springer Series in Information Sciences, vol 8, Springer Verlag, Berlin , 1987, pp 16-21.

[Kohonen 87b] T. Kohonen, Self-Organization and Associative Memory, second edition, Springer Series in Information Sciences, vol 8, Springer Verlag, Berlin , 1987, pp 261-263.

[Kohonen 88] T. Kohonen, "An Introduction to Neural Computing," Neural Networks : The Official Journal of the International Neural Network Society, Pergamon Press, Vol.1, pages 3-16, 1988.

[le Cun 87a] Y. le Cun, "Modèles Connexionnistes de l'Apprentissage," Thèse de Doctorat, Université Pierre et Marie Curie, Paris, juin 1987, pages 155-162.

[le Cun 87b] Y. le Cun, "Modèles Connexionnistes de l'Apprentissage," Thèse de Doctorat, Université Pierre et Marie Curie, Paris, juin 1987.

[le Cun 87c] Y. le Cun, "Modèles Connexionnistes de l'Apprentissage," Thèse de Doctorat, Université Pierre et Marie Curie, Paris, juin 1987, pages 126-131.

[Liu 89] Y. D Liu, G. Z. Sun, H. H. Chen, Y.C. Lee, "Grammatical inference and neural network state machines," Proceedings of IJCNN 89, Washington D.C., USA, January 1990.

[Lippmann 87] R. Lippmann, "An introduction to computing with neural nets," IEEE ASSP Magazine, April 1987.

[McClelland 88] J. L. Mc Clelland and D. E. Rumelhart, "Explorations in Parallel Distributed Processing, a Handbook of Models, programs, and Exercices," MIT Press, Cambridge, Massachussets, London, England 1988.

[Minsky 88] M. L. Minsky and S. A. Paper, "Perceptrons," Expanded Edition, MIT Press, 1988.

[Park 88] K. Park, "Sequential Learning : Observation on the Internal Code Generation Problem," Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton, T. Sejnowsky, Morgan Kaufmann Publishers, 1988.

[Pearlmutter 88] B. A. Pearlmutter, "Learning State Space Trajectories in Recurrent Neural Networks," CMU-CS-88-191, December 31, 1988.

[Pineda 87] F. J. Pineda, "Generalization of Back-Propagation to Recurrent Neural Networks," Physical Review Letters, Vol. 59, Number 19, 9 November 1987.

[Pineda 89] F. J. Pineda, "Recurrent Back-Propagation and the Dynamical Approach to Adaptative Neural Computation," Neural Computation 1, 161-172, 1989.

[Rumelhart 86] D. E. Rumelhart, G. E. Hinton, and R.J Williams, "Learning internal representations by error propagation," dans "Parallel Distributed Processing : Explorations in the Microstructure of Cognition," Vol 1 : Foundations, MIT Press, Cambridge, MA, 1986.

[Rohwer 88] R. Rohwer and S. Renals, "Training Recurrent Networks," Proceedings of nEURO 88, Personnaz L. and Dreyfus G. Eds, ESPCI, Paris, France, June 6-9, 1988.

[Schreter 88] Z. Schreter, "Short-term memory/Long-term memory interactions in connectionist simulations of psychological experiments on list learning," Proceedings of nEURO 88, Personnaz L. and Dreyfus G. Eds, ESPCI, Paris, France, June 6-9, 1988.

[Sejnowski 86] T. Sejnowski and C. Rosenberg, "NETtalk : a parallel network that learns to read aloud," The Johns Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01, 32pp., 1986.

[Servan-Schreiber 88] D. Servan-Schreiber, A. Cleeremans and J. L. McClelland, "Encoding sequential structure in simple recurrent networks," Technical Report, Carnegie Mellon University, CMU-CS-88-183, November 1988, 37 pages.

[Smith 89] A. W. Smith and D. Zipser, "Encoding sequential structure: experience with real-time recurrent learning algorithm," Proceedings of IJCNN 89, Washington D.C., USA, June 1989.

[Todd 88] P. Todd, "A Sequential Network Design for Musical Application," Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton, T. Sejnowsky, Morgan Kaufmann Publishers, 1988.

[Touzet 88] C. Touzet et N. Giambiasi, "Reconnaissance de séquences par des réseaux de neurones," Neuro-Nîmes 88, Nîmes, France, novembre 1988, pp 383-394.

[Touzet 89a] C. Touzet et N. Giambiasi, "Neuromimetic Sequential Machines," IASTED expert systems, Zurich, Suisse, 26-28 juin 1989.

[Touzet 89b] C. Touzet et N. Giambiasi, SACREN, rapport final du contrat ANVAR N° A8801006 JAL, juillet 1989.

[Touzet 90a] C. Touzet et N. Giambiasi, "Connectionist finite-state machines," Proceedings of IJCNN 1990, Washington D.C., USA, 15-19 janvier 1990.

[Touzet 90b] C. Touzet et N. Giambiasi, "Quelques approches neuromimétiques pour le traitement des séquences," NSI 90, Centre Paul Langevin, Aussois, 7-10 mai 1990.

[Tsung 89] Fu-Sheng Tsung and G. W. Cottrell, "A sequential adder using recurrent networks," Proceedings of IJCNN 89, Washington D.C., USA, June 1989.

[Watrous 89] R. L. Watrous, "Phoneme discrimination using connectionist networks," submitted to the Journal of the Acoustical Society of America, May 2, 1989.

[Williams 88] R.J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," ICS Report 8805, Univ. of California, San Diego 92093, October 1988.